AMSTRAD USER

AÑO I - Número 1 - Octubre 1985 - 300 ptas.



PAGINA TRES

Que la informática ha revolucionado nuestras vidas no es nada nuevo. Los microordenadores personales han pasado, en cinco años, de ser unos artículos desconocidos por el gran público a convertirse en el más vendido.

Una empresa apuesta firmemente por el futuro: Amstrad. Los productos que lanza constituyen ejemplos de ingeniería bien hecha, con componentes avanzados que confieren características espectaculares y a unos precios tan competitivos que ponen en apuros incluso a los fabricantes que antes eran líderes del mercado. La relación calidad/precio que ofrecen es la clave del éxito.

Una observación atenta del mercado de los productos Amstrad nos revela que la mayoría de la gente se ha comprado el ordenador como instrumento de utilidad en casa o en el trabajo con el que poder aprender y trabajar mejor. Esto no quiere decir que ésta sea una revista dedicada exclusivamente a tratar temas de contabilidad, pero tampoco es un simple compendio de juegos para pasar los ratos aburridos. Vamos a ofrecer una información amplia respecto a todos los temas que abarca la informática. Juegos, gestión de empresas (campo en el que Amstrad desbanca a ordenadores mucho más caros), gestión doméstica y, naturalmente, el tema educativo que poco a poco va surgiendo en nuestro país. Basta adelantar que ya son muchos los colegios que utilizan este ordenador.

En este primer número que tiene en sus manos, podrá encontrar un conjunto de temas que le serán útiles. El tema central se refiere a la revolución de los discos. Algo que hasta hace poco se consideraba un lujo de ricos y que se impone por sus ventajas sobre el cassette. También hay listados de programas para los que quieran teclear y el «Taller de Programación», en el que se irá explicando cómo usar mejor diversas características del BASIC.

Otra sección irá examinando otros lenguajes existentes para la máquina. Para aquellos que prefieren usar programas hechos tenemos secciones dedicadas a analizar programas comerciales.

Por último queda comentarles dos apartados más directamente relacionados con usted. El primero es el del correo; intentamos ayudarle en sus problemas y recibir sus sugerencias. La otra sección, Usuario, nos mostrará los diversos usos a los que se puede destinar la máquina. Para estrenarla tenemos a Joan Guillén, un humorista gráfico de Barcelona, que se dedica a dibujar con su ordenador.

Naturalmente, estamos abiertos a todas sus sugerencias. Esperamos que disfruten con Amstrad User.

Director
Santiago Gala

Subdirector
J. A. Sanz

Redacción
Francisco Martín
Teresa Rubio

Diseño
Rodrigo López Alonso

Portada
Bravo Lofish

Edita
Indescomp, S. A.

SERVICIO AL CLIENTE
Conchita García
Tel. (91) 933 45 48

Realización y Coordinación
Publinformática, S. A.

Jefe de publicidad
María José Martín

Dirección y redacción
Bravo Murillo, 377 5.° A
Tel. 733 74 13
28020 Madrid

Publicidad y Administración
Bravo Murillo, 377 5.° A
Tel. 733 96 62/96
28020 Madrid

Publicidad Madrid
Tel. 733 96 62

Publicidad Barcelona
Tel. 301 47 00 Ext. 27/28
y 318 02 89

Depósito legal
M-32.038-1985

Distribuye
S.G.E.L.
Avd. Valdelaparra, s/n
Alcobendas (Madrid)

Fotocomposición
Cicegraf
Carare, 6 (posterior)
28033 Madrid

Fotomecánica
Karmat
Pantoja, 10
28002 Madrid

Imprime
Gráficas Velasco
Antonio Cabezón, 10
28034 Madrid

AMSTRAD USER







por fin, el último comentario es Decathlon, un juego para hacer deporte sin sudor.

Escritores eléctricos

Uno de los juegos más interesantes y mejor presentados que han salido para el Amstrad 464 es el *Sorcery* de Virgin Games. En este artículo se cuentan algunos secretos de su realización.

Muchas compañías con dinero piensan que pueden tener éxito en el mercado de los programas para ordenadores caseros; después de todo, lo único que hace falta es un buen producto y propaganda. Inicialmente, Virgin cayó en esta trampa y publicó lo más decente de todos los programas que recibió en respuesta a un anuncio que apareció en las revistas inglesas.

De resultas de ello, aprendieron que la mejor propaganda es la que se realiza de boca de un amigo a oído de otro, y volvieron a realizar otro ataque justo cuando **Virgin** volvía a subir hacia arriba. El resultado de ese encuentro es *Sorcery*, uno de los mejores juegos que hay para este ordenador.

Crecimiento veloz

La sede central de Virgin está al lado del popular «Portobello Road» de Londres, rodeada de tiendas y mercados callejeros. En una calle pequeña y mal cuidada se puede ver detrás de unos andamios un cartel escrito a mano que dice «Virgin por aquí...» Dentro se huele a casa recién terminada, a yeso y pintura fresca. Desde que Virgin empezó como una compañía de discos (con músicos como **Mike Oldfield**) ha crecido y los departamentos de discos, películas, vídeos y líneas aéreas se han trasladado a un edificio cercano.



La pantalla nos muestra unos dibujos de gran calidad junto con un texto fácilmente legible.

Los programadores están en otro edificio más alejado, detrás de una elegante puerta de estilo antiguo se encuentra una recepcionista sueca, un télex y más albañiles. Una habitación tiene vida propia, la de los ordenadores. Todo tiene un aspecto revuelto y desordenado.

El QL está en un rincón cubierto de polvo. En este sitio se escribio el *Sorcery*

En el programa se ha empleado un montón de trabajo, y esto se ve desde el momento en que empieza a leerse de cinta. En la carta se ha utilizado un sistema denominado *Flashload*, un cargador de alta velociad. Todos los nuevos juegos de Virgin utilizan un sistema parecido, ya que resultaba esencial en algunos ordenadores de carga muy lenta (como el Commodore 64 que tiene una velocidad tres veces más lenta que el Amstrad). De modo que aunque no era muy necesario para el Amstrad, se decidió que los 2 K de memoria usados por el *buffer* de cinta eran demasiado valiosos para desperdiciarlos y que había que hacer un cargador especial. Esto causó muchos problemas y sólo funcionó bien al final. La idea original era cargar a 4.000 baudios de velocidad (el cuádruple de la velocidad normal), pero mientras que era válido en las pruebas que hicieron, se comprobó que daba demasiados errores al fabricar las cintas en masa. De modo que se fue reduciendo la velocidad poco a poco hasta que se consiguió una que no



La presentación del juego nos muestra algunos de los peligros a los que nos tendremos que enfrentar.

diese errores. La cinta comercial tiene una velocidad de 250 baudios pero sin espacios entre bloques. Incluso así se tarda bastante en cargar debido a la longitud del programa. La posibilidad de una versión en disco es interesante, sobre todo ahora que está alcanzando una gran difusión. Pero durante el desarrollo del juego todavía no lo tenían y todo el sistema se desarrollo en un 464 con cassette y el *Devpac* de **Hisoft.**

Todo desarrollo en un Amstrad

Los sistemas de desarrollo son una cuestión de gusto. Algunas compañías como Abersoft (que hizo el Forth) comparten grandes sistemas, algunas otras utilizan miniordenadores y otras microordenadores más potentes como el Apricot, cada programador elige el que más le gusta en función de diversos aspectos, como su velocidad o la cantidad de utilidades que tiene para escribir programas. Normalmente se usa un cable para transmitir el programa final a la máquina donde se va a ejecutar.

Virgin es especial en esto, ya que es una de las pocas casas que hacen todo el trabajo en la máquina donde se va a ejecutar. A **Dave**, que ha escrito la mayor parte del *Sorcery*, le gusta este sistema, pero en cambio **Andy** se queja continuamente del Commodore 64 con el que está trabajando y desearía tener algunos discos más rápidos. Otro punto importante son las utilidades y en este apartado, aparte del *Devpac 2*, se ha creado en **Virgin** un programa para crear caracteres multicolores. Todas las habitaciones fueron creadas por **Ian**, que usó solamente 256 caracteres para todo.

Trucos mágicos

Si ha visto el juego, habrá notado que la parte superior de la pantalla aparece en modo 0 mientras que la parte inferior está en modo 1. Esto se consigue esperando a que el haz del monitor llegue a la mitad de la pantalla y cambiando entonces de modo. Así se consigue una gran versatilidad con los gráficos ya que se obtienen 16 colores en la parte superior y un texto decente en la inferior. El libro que hace de reloj desaparece de cuatro en cuatro puntos, ya que éste era el sistema más fácil de escribir el programa, al corresponder cada cuatro puntos a un byte.

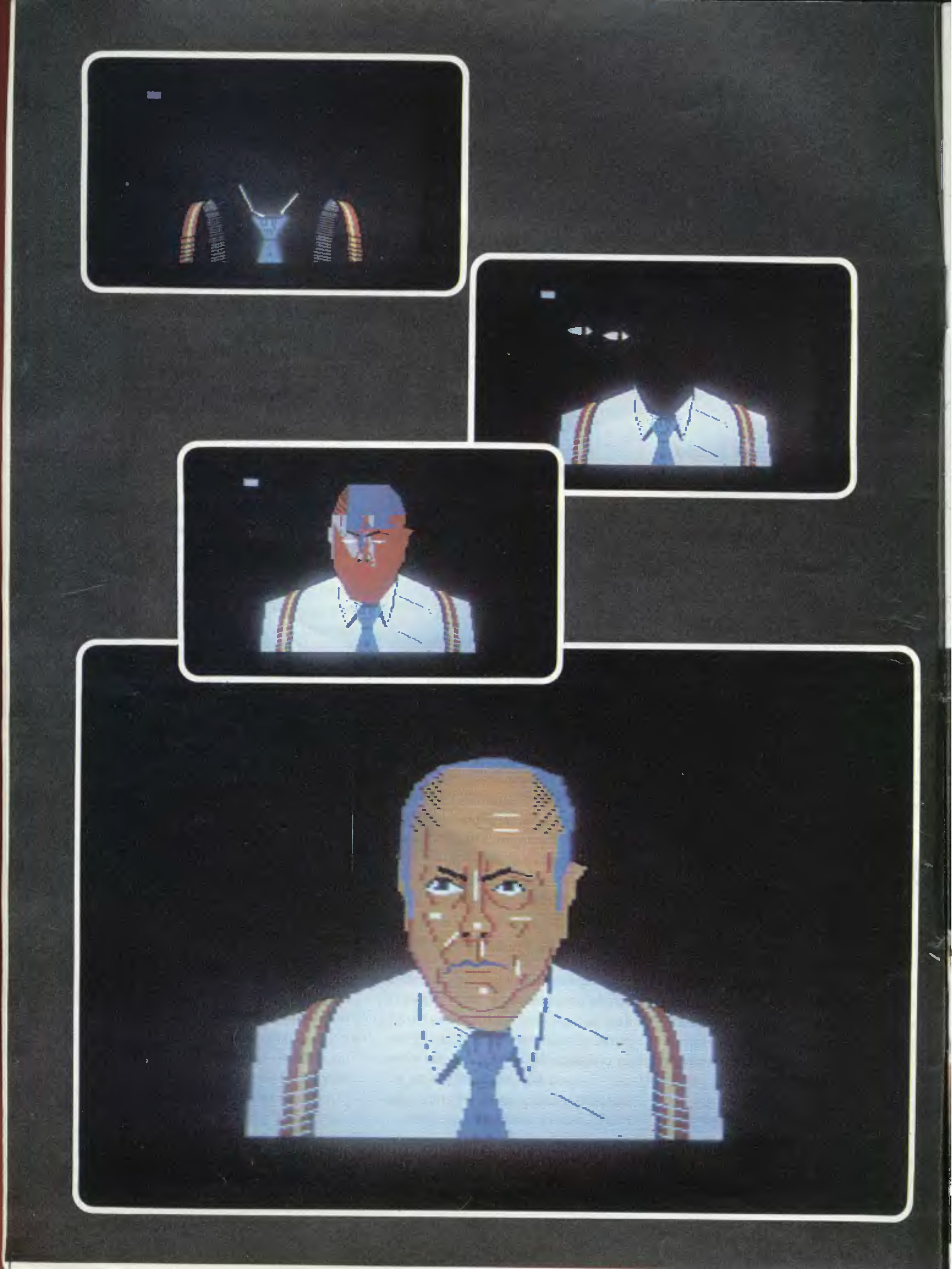
¡Un error!

Hay un error, pequeño, en el programa. Cuando el hechicero vuela detrás de una catarata, a veces deja una copia de sí mismo en el agua. Inicialmente **Dave** atribuyó esto a la «refracción de los líquidos», pero posteriormente reconoció que se debía al modo en que la rutina de movimiento de sprites trabajaba usando el *XOR* (instrucción en lenguaje máquina); asimismo señaló que no hay habitaciones donde dos monstruos del mismo tipo se puedan cruzar, ya que esto los haría desaparecer. Hasta ahora él había programado el **Spectrum** y ésta era la primera vez en que podía jugar con tantos colores. Ian realmente se esmeró con los gráficos y hay multitud de detalles en los que fijarse, como qué Cauldrones son buenos y cuáles malos dependiendo de la posición inicial del juego. También hay un mensaje especial si su nombre es «Hugh» y se consigue la máxima puntuación. Todas estas pequeñas cosas mejoran el juego.

El futuro

Hay dos proyectos en consideración para el futuro. El primero se llama *Strangeloop*, basado en un gran éxito del **Spectrum**, y el segundo es parecido al *Sorcerer* pero todavía sin nombre ni guión. Para la casa es importante mantener la reputación de calidad conseguida con el **Amstrad** y es importante que todos los demás programas sean buenos. Todos los nuevos programnas tendrán gran calidad de gráficos y un mejor sonido, en estéreo seguramente.

De todos modos, **Virgin** es un nombre que hay que tener en cuenta de cara al futuro.



Joan Guillén

MI LAPIZ ES UN AMSTRAD



*El estudio de **Joan Guillén** resume su trabajo en una aparente contradicción: una gran muñeca de cartón cuelga de la pared, mirando muy seriamente hacia el ordenador, un Amstrad CPC-464 que, por una vez, pierde protagonismo ante las manos que lo manejan. Más allá, maquetas de cartón piedra, dibujos y el aplastante calor de Barcelona.*









Joan Guillén

El nombre de Joan Guillén es muy conocido en los medios periodísticos y gráficos de Barcelona: ha realizado caricaturas y fotomontajes satíricos en **Destino, Por Favor** y **La Calle,** entre otras; una faceta menos conocida es su investigación como profesor de Escenografía en el **Instituto del Teatro:** ha realizado, por ejemplo, el montaje escénico del espectáculo de **La Trinca** que se presentará en breve. Hace algo más de un año se sentó frente a un ordenador a ver qué se podía hacer.

«Al principio pensé en trabajar con un CINEKIN, esos aparatitos con una banda de papel vegetal, que giraban y reproducían dibujos animados, pero tenía problemas de sincronismo. Entonces pensé que un aparato electrónico podía funcionar mejor. No había tenido ninguna relación anterior con la informática, ni siquiera me gusta jugar a las máquinas de marcianos de los bares. Comencé con un Spectrum, y me convencí de que se podia hacer mucho, pero el ordenador tenía unas limitaciones claras.

Estuve buscando otro aparato, pero antes de pasar a máquinas grandes, quise quedarme en un tamaño intermedio, porque yo me dedico al periodismo, al grafismo inmediato. El ordenador doméstico tiene una gran flexibilidad de lenguaje y, además, resulta muy cercano al usuario. El Amstrad es una máquina increíble en este aspecto, muy rápida, y que permite trabajar en condiciones satisfactorias.»

No utiliza ningún tipo de periférico de dibujo, ya que le interesa «programar» la aparición de la imagén, y no el dibujo «a mano alzada».

El ordenador no es más que un medio de trabajo

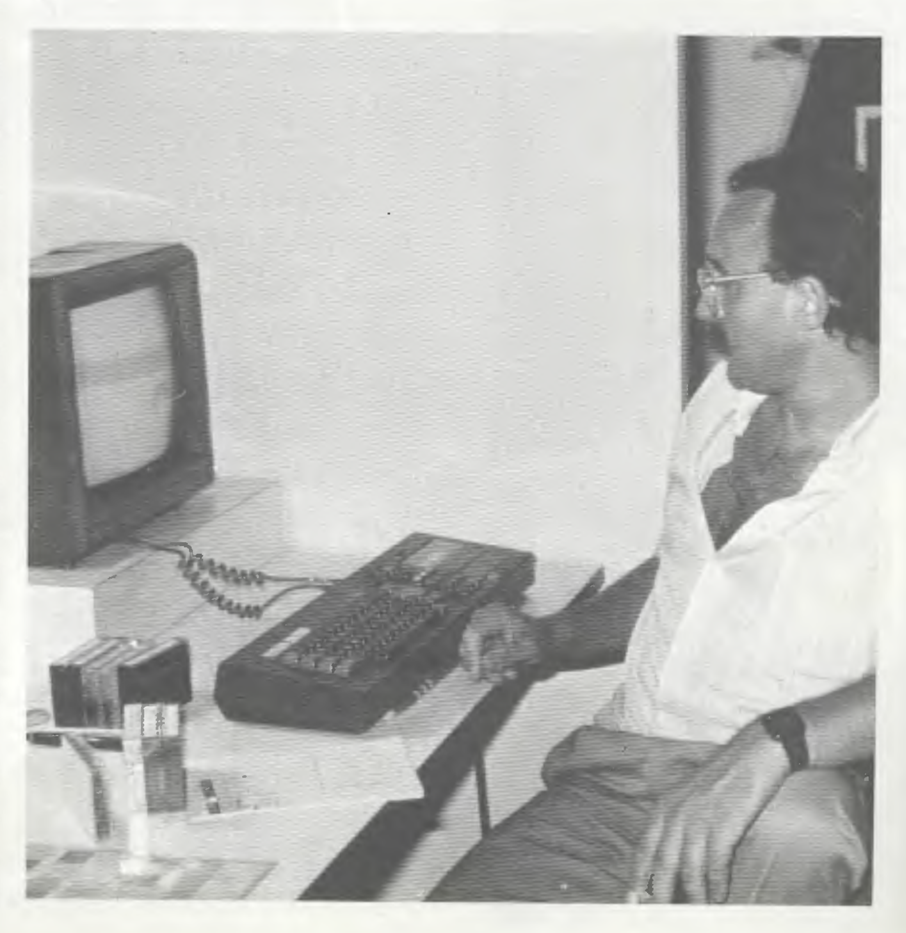
«Si yo utilizo un ordenador de cinco o diez millones, el resultado es demasiado perfecto, y el espectador piensa: Claro, con ese aparato cualquiera hace eso.» En cambio, cuando contemplamos los **microclips**, la mano de **Guillén** se distingue entre las líneas del monitor de su Amstrad.

«He tenido grandes problemas, porque mi trabajo es para TV3, y no es fácil obtener una salida de vídeo en las condiciones que piden los técnicos. De momento filmamos la imágen, pero el sistema no es satisfactorio».

Nos cuenta que, dos veces por semana, aparece por su estudio un equipo de filmación, que toma en vídeo la secuencia. A continuación se añade la banda sonora en el estudio de edición. Los **microclips** aparecen en el telediario del mediodía, satirizando o comentando alguna noticia de actualidad.

Otro problema con el que se enfrenta es la realización de copias en papel: no existe todavía ningun programa de copia para impresoras en color. En cuanto a la ayuda técnica que le puede brindar el distribuidor, está un poco desencantado: «Cuando ven que estás realizando un trabajo profesional, todos se brindan a ayudarte, pero luego te quedas esperando, y... hasta ahora. Yo no soy informático: soy un dibujante de prensa, y hay problemas técnicos que no me interesan.»

Joan dibuja en BASIC, porque así es consciente del proceso de síntesis de la imágen. Utiliza como mucho algunas llamadas a rutinas en código máquina, pero no está interesado en los detalles. Construye las figuras mediante formas geométricas, buscando más el efecto dramático que la perfección técnica: algo muy cercano a la caricatura. Sin embargo, el dinamismo de la imágen aleja su técnica del **comic**. Nos enseña alguno de los más recientes, y el resultado salta a la vis-





ta. Unas imágenes vivas, que saltan de la pantalla y se mueven ante nosotros.

El teatro, otro terreno de investigación.

La escenografía es otro campo donde le pareció interesante el uso del ordenador, sobre todo en sus aspectos de simulación: «Estoy trabajando en un programa que permita diseñar y visualizar inmediatamente espacios escénicos, con la comodidad que implica no tener que realizar maquetas en volumen. Mediante él se facilita la experimentación y el trabajo creativo.»

Otro aspecto interesante es la integración del ordenador como elemento activo de un montaje. «En el último espectáculo de **La Trinca** hemos utilizado un Amstrad con una pantalla de cuatro por tres metros, en blanco y negro. Con una sincronización adecuada, el ordenador dibuja fondos y personajes mientras ellos cantan la canción. La facilidad de controlar mediante el ordenador un juego de luces o determinados elementos escénicos

«Lo importante no es el lenguaje del ordenador: si la idea no vale la pena, no la salva ninguna tecnología.»

la he dejado para más adelante, aunque no renuncio a este tipo de trabajo.»

«Durante la segunda mitad de septiembre estaré en un Encuentro sobre Teatro en Sevilla, en la Universidad Menéndez y Pelayo. En él quiero mostrar lo que llevo trabajado sobre el uso del ordenador como elemento de diseño escénico. Estoy preparando una exposición en Barcelona, con obras realizadas por ordenador.» Para este tipo de trabajo, **Joan** saca por impresora el dibujo y le da un acabado «a mano», usando fotocopia y añadiendo trazos a mano que **rompen** la imágen. «El resultado es cercano a la serigrafía o litografía.»

Charlamos en su estudio y después en una cafetería. La sensación que nos da es que **Joan Guillén** es un entusiasta de su trabajo, y que no puede parar. Nos comenta las incomprensiones de algunos compañeros: «En un congreso de historietistas, hubo quien puso el grito en el cielo. Me decían que me había pasado a "las maquinitas",... No se trata de máquinas o no: me dedico a ésto porque está ahí, y brinda muchas posibilidades. Por otro lado, me han propuesto pasar a trabajar con un sistema de diseño gráfico, pero tengo miedo de perderme cada vez más en aspectos técnicos. Estoy escribiendo un libro, con el que pretendo ayudar a los usuarios de ordenadores domésticos a crearse un estilo propio, a dibujar de una manera creativa.»

Y salimos al bochorno de la calle, con la sensación de que **Joan Guillén** ha encontrado ese difícil punto de equilibrio entre artesanía y tecnología punta, entre la repetición industrial y la creación artística. Y mientras paseamos por la Diagonal de Barcelona sus **microclips** nos siguen rondando por la cabeza.





CPC 664

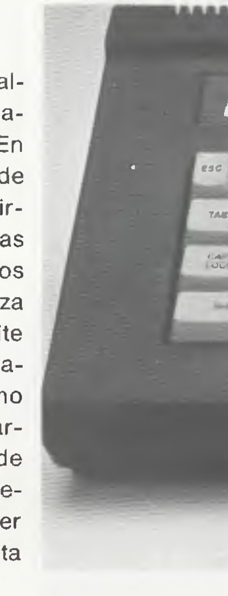
La revolución de los discos

Con la aparición de las unidades de disco para el CPC 464 y del nuevo CPC 664, se ha producido un avance cualitativo en la línea de productos Amstrad.



Si el primer modelo incorporaba características realmente interesantes e innovadoras, la introducción de los discos hacen que se pueda convertir un ordenador que estaba dirigido al mercado doméstico en una máquina perfectamente válida para el campo empresarial. Todo el mundo ha oído comentar las excelencias de los discos y sabe que son mejores que las cintas, pero ¿por qué?, ¿qué mejoras ofrecen?, ¿le interesan a usted realmente? Estas son las incógnitas que vamos a intentar resolver.





Cómo se graba la información

Tanto la cinta como el disco utilizan un sistema magnético de grabación. Este, básicamente, es un soporte muy fino, de plástico, recubierto de partículas de algún óxido metálico. Estas partículas pueden orientarse de un modo u otro, según se les someta a un campo magnético u otro. Para ello se dispone de una cabeza grabadora/lectora que puede generar ambos campos magnéticos según se les proporcione una corriente eléctrica u otra. Además esta cabeza puede funcionar, como su nombre indica, para leer las orientaciones (que representan datos) que previamente se han grabado.

Hasta aquí ambos sistemas son iguales, ya que el ordenador le envía al sistema de grabación los datos que son convertidos en señales eléctricas y éstas a su vez se transforman en señales magnéticas en el disco o cinta. Las diferencias surgen en la disposición de la superficie magnética y en la unidad lectora. En un cassette, la información se graba sobre una cinta de bobina que pasa lentamente delante de la cabeza, y si se quiere acceder a un programa (o datos) grabado al final, deberán pasar antes todos los demás. Además este sistema, debido a su constitución, no permite manejar ficheros de acceso aleatorio (aquellos en los que se puede acceder a cualquier parte del fichero instantáneamente como si fuese una matriz). En cambio, en un disco, el sistema de grabación que se utiliza se basa en circunferencias concéntricas llamadas pistas y no en una espiral como en los de música. Esto, unido a que la cabeza dispone de un motor que le permite colocarse en cualquier pista rápidamente, hace que el acceso sea mucho más veloz que en la cinta. Por otra parte, la velocidad de giro del disco es de unas trescientas revoluciones por segundo, lo que permite que cualquier sección de una pista sea leída o escrita rápidamente.

Directorios

Otra diferencia importante entre las cintas y los discos es cómo maneja el ordenador cada uno de los dispositivos. En la cinta, cuando le mandamos grabar, lo hace en el lugar en que ésta se halle, borrando lo que hubiese debajo sin comprobar qué es. Por lo tanto, queda de nuestra responsabilidad el evitar borrar ficheros accidentalmente. Otro detalle es que el índice del contenido de cada cinta se debe apuntar por separado, a menos que cada vez que queramos ver qué tiene hagamos pasar toda, lo que puede llevarnos bastantes minutos.

Estos problemas se ven solucionados de un modo muy eficaz con el disco. En primer lugar, y debido a que es el propio ordenador el que controla los motores de la cabeza y de rotación, se encarga él mismo de elegir el sitio donde va a grabar (por medio del directorio que veremos posteriormente) y no existe el problema de que borre otro archivo a menos que se lo indiquemos expresamente. El directorio es una parte específica del disco (siempre la misma en todos, para que el ordenador sepa dónde ir a buscarla) en la que se indica qué programas hay en él y dónde están situados. De este modo se comprueba qué sitios hay libres para grabar y se puede obtener instantáneamente el índice del contenido del disco.



El CPC 664 ha sido el primer ordenador que ha introducido la unidad de disco incorporada a un precio asequible, lo que representa un avance espectacular sobre las anteriores tendencias de la microinformática.

Los discos en el Amstrad

Hasta ahora hemos hecho una descripción genérica de los discos y su funcionamiento. Aunque esto es válido también para los discos del Amstrad, hay muchas características propias que vamos a ver ahora. Las unidades encargadas de leer y grabar, denominadas popularmente *drives*, poseen dos formatos distintos. En el CPC 464 se venden aparte y tienen una forma alargada. En cambio, en el Amstrad CPC 664 el *cassette* se ha quitado y ha sido sustituido por un disco; se puede conectar otro exteriormente, situado en la parte derecha y que sobresale ligeramente del resto del aparato. En ambos la carga se hace por la parte frontal y para expulsarlo hay que pulsar un botón.

El formato de los discos es de tres pulgadas y se presentan en un estuche rectangular duro (aunque los discos se denominen flexibles) que, contrariamente a lo que se podía pensar, no hay que abrir, sino que se introduce tal cual dentro del lector. Dentro de este estuche se encuentra el disco propiamente dicho que es leído y escrito por medio de una ranura oval doble (una por cada lado) que normalmente se encuentra cubierta por una trama metálicaa que se retira automáticamente al introducirla en el lector.

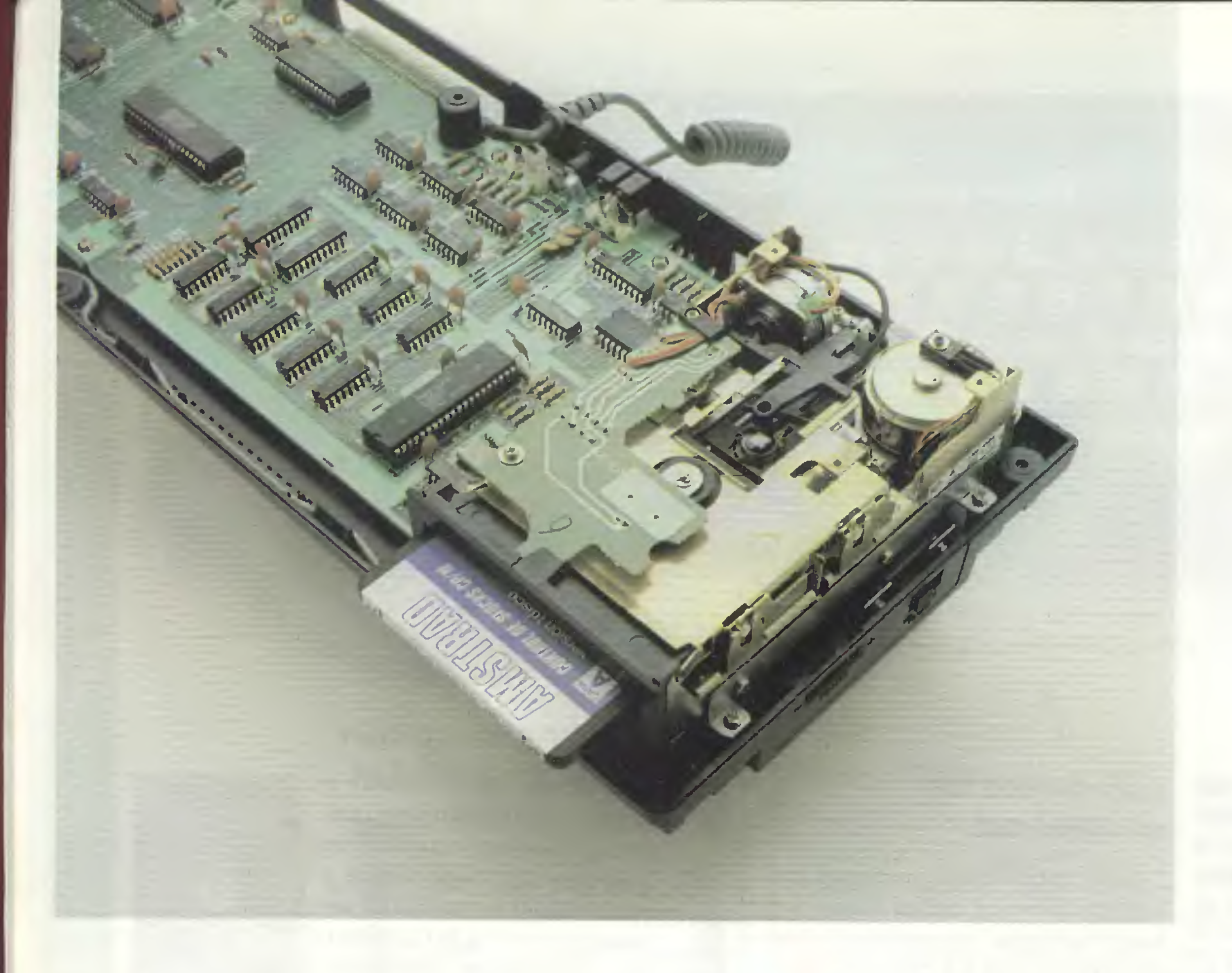


El disco, al igual que las cintas y al contrario que el resto de los formatos de discos usados en otros ordenadores, puede leerse y escribirse por las dos caras, según se introduzca por un lado u otro y dispone de unas piezas deslizables (una por cada lado) que según en la posición en que se encuentren dejan escribir en el disco o sólo dejan leer.

El sistema utilizado internamente para grabar la información es de cuarenta pistas (circunferencias concéntricas) y en cada una de ellas caben nueve sectores (divisiones de la pista), excepto en un formato especial denominado IBM (que no se suele usar), con una capacidad en cada uno de los 512 bytes. Si se va a usar el CP/M, las dos primeras pistas las emplea el sistema para almacenar el núcleo de este sistema opertivo, por lo que la capacidad real disminuye en nueve Kilobytes. En cambio, si un disco va a ser usado solamente para guardar programas en BASIC, no hace falta reservar estas dos pistas y se dispone del máximo espacio.

Los sistemas operativos.

Un sistema operativo es un programa especial que se encarga de con trolar todas las operaciones: lectura,



La construcción de la unidad de discos es robusta y fiable. Se puede ver al fondo, a la derecha, el motor que mueve el disco, y en el centro el eje que sujeta el disco, encima del cual se halla la cabeza de grabación/lectura.

escritura de programas y ficheros, de modo que se realicen de un modo rápido y eficaz, evitándole al usuario un montón de operaciones fastidiosas, ya que actualiza automáticamente el directorio, comprueba que no se cometan errores, etc.

Normalmente, un ordenador puede funcionar con uno o varios sistemas operativos, y cada uno es uítil para una misión específica. El Amstrad dispone de dos distintos: el **AMSDOS** y el **CP/M.** El **AMSDOS** ha sido desarrollado por la misma casa y constituye el soporte básico de funcionamiento de la máquina, pero no es suficiente para poder sacar todo el provecho a los discos, por lo que es necesario usar el **CP/M** como complemento. Este es un sistema operativo estándar diseñado para que pueda funcionar en cualquier ordenador que disponga de unas características mínimas existentes en el Amstrad. El hecho de que sea estándar implica que dispone de una librería de programas muy amplia debido a la popularidad del sistema operativo.

Con el ordenador se suministran ambos sistemas, incluyendo las utilidades básicas del sistema como es la copia de discos y programas similares. El **AMSDOS** es el que se instala al encender la máquina. Si se quiere utilizar el **CP/M** se debe introducir el disco correspondiente y teclear «|cpm», con lo que el ordenador nos pasará automáticamente a ese sistema operativo y nos mostrará el cursor del **CP/M «A>»**.

Además se suministra con los discos el lenguaje Logo (del que publicamos un artículo en este mismo número), baste decir aquí que es un lenguaje pensado para enseñar informática a los niños y con unas características que le dan una gran potencia, superior en algunos casos al BASIC que también trae la máquina

Documentación

Dicen las malas lenguas que los manuales de los ordenadores se hacen mal a propósito porque tienen un acuerdo con las casas editoras de li-

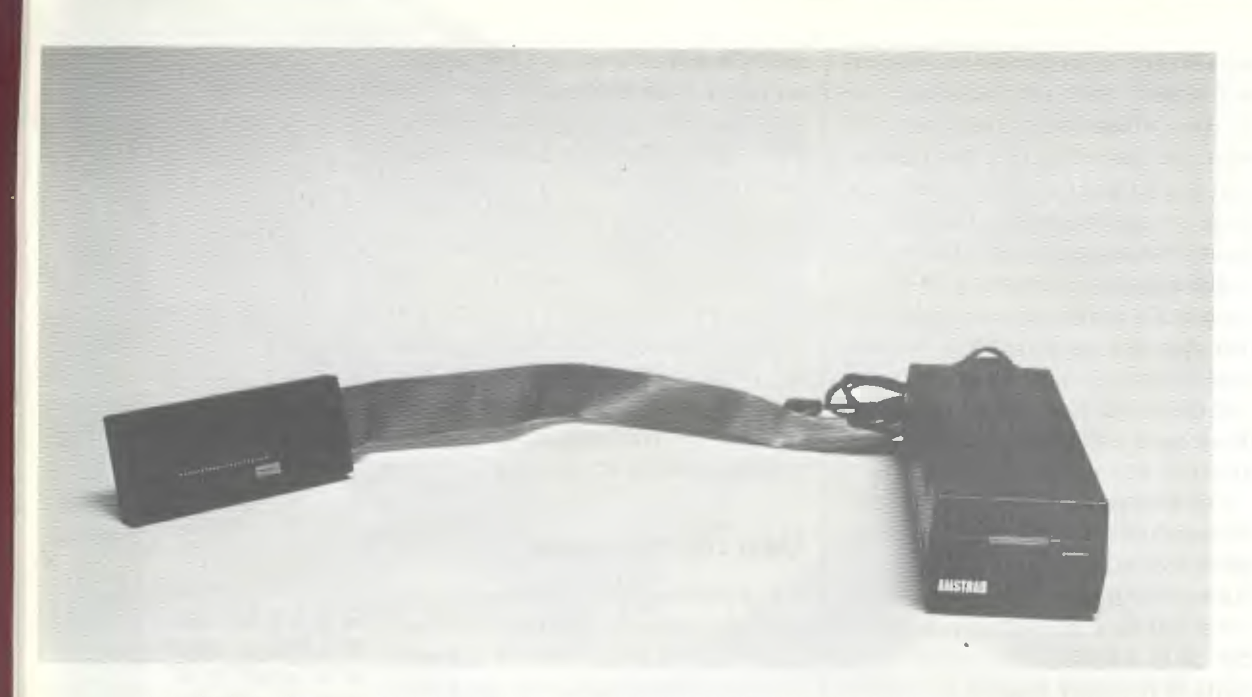
bros. Nosotros no somos tan mal pensados, pero hemos de reconocer que no conocemos ningún ordenador del mercado que traiga una documentación que agrade a todo el mundo. Los usuarios que empiezan a manejar ordenadores suelen opinar que los manuales son muy complejos y no los entienden. En cambio los veteranos opinan que las documentaciones son muy banales y que deberían incluir más datos sobre el funcionamiento interno de la máquina, datos que confundirían aún más al usuario novel.

Con esta situación resulta muy difícil hacer un manual al gusto de todos, sobre todo si se considera que los manuales del Amstrad son ya gruesos de por sí, debido a las muchas posibilidades de la máquina. Por eso, el fabricante ha preferido situarse en un nivel medio, pensado (y no con poca razón) que rápidamente saldrán libros para cubrir los huecos existentes. Entre los diversos capítulos sobre las unidades de disco y los programas que se incluyen, hay uno dedicado al **AMSDOS** y al **CP/M**, otro al Logo, apéndices y resúmenes de referencia. Todos ellos están pensados para un usuario con cierta experiencia en la máquina y que por tanto ya domina algo el tema. El desarrollo de las explicaciones es corto y conciso, incluyendo algunos ejemplos que explican el uso de todos los comandos del AMSDOS y de los más importantes del CP/M (los manuales del CP/M editados por **Digital Research,** que es la casa que proporciona este sistema operativo, son bastante más densos y complejos).

Uso del sistema

El funcionamiento del sistema resulta sencillo si sólo se desean utilizar programas llave en mano, ya que normalmente lo único que hay que hacer es teclear «*run* "nombre del programa"» para que se ponga en marcha, y ya se encarga éste de realizar todas las operaciones necesarias. Realmente se hace notar el aumento de velocidad comparado con la cinta. Re-

En el CPC 664 todo el sistema de control se ha incluido en la placa del ordenador. Asimismo, se ha dejado un conector en la parte trasera derecha para conectar una segunda unidad.



La unidad de disco externa se puede conectar al CPC 464 o como segunda unidad del 664. Su pequeño tamaño no es óbice para que el funcionamiento sea pefecto.

sulta muy difícil acostumbrarse a la cinta una vez que se ha usado el disco, y no sólo por la velocidad, también está la facilidad de manejo y el no tener que acordarse en qué cinta está cada programa para no borrarlo.

Con respecto a la fiabilidad, resulta asombrosa, ya que parecería que un sistema más rápido tendería a fallar más, pero hay que tener en cuenta que el disco se ha desarrollado especialmente para los ordenadores, no como las cintas que ya existían y fueron posteriormente adaptadas para este uso. Durante el tiempo que lo estuvimos usando no se produjo ningún error y todo funcionó perfectamente.

Cuando el usuario quiere emplear los discos para guardar sus propios programas y datos, el uso resulta algo más complicado en un principio, ya que el BASIC sólo es accesible desde AMSDOS y en cambio diversas utilidades (como es la del formateado de discos) sólo están disponibles en CP/M, por lo que hay que estar cambiando entre uno y otro. No obstante, al final se entiende y no crea ningún problema grave.

Otro detalle muy importante es la capacidad de los discos. En cada cara caben 160 Kbytes, es decir, 160.000 bytes aproximadamente (equivalentes a un carácter por byte si se van a almacenar textos), que es más del doble de la memoria que tiene instalada el ordenador.

Comandos disponibles

En principio, todos los comandos disponibles para la unidad de cinta se pueden usar también con los discos. Así, «ACT» nos dice todos los programas que hay en un disco (sean de **CP/M** o de **AMSDOS**, ya que el formato de ambos es similar) pero a una velocidad considerablemente superior a la de la cinta. Asimismo, los comandos SAVE, LOAD, CHAIN, MERGE, PRINT #9, INPUT #9, etc., también se pueden usar con el disco, pero siempre que hagamos referencia a un fichero (*OPEN, SAVE, LOAD,* etc.) se deberá poner obligatoriamente el nombre de éste detrás, mientras que con la cinta era opcional, y si no se indicaba cargaba el primero que encontraba, pero al tener los discos una disposición distinta este valor, por defecto, no es válido.

Además de estos comandos existe otro conjunto de órdenes pensadas específicamente para los discos. Todas ellas se distinguen porque van precedidas de la barra vertical (|) seguida del nombre del comando sin espacio en medio. De este modo resulta fácil recordarlas y se evitan equivocaciones. Estos comandos amplían las posibilidades de los anteriores y permiten seleccionar una de las dos uni-

dades de disco (cuando tenemos conectada más de una), cambiar de nombre a los ficheros, borrarlos y entrar en CP/M. Desde el BASIC el manejo es muy similar a como se hace en cinta, y el sistema de creación de ficheros se programa igual que si se hiciese en cinta, pero cuando se ejecuta el programa no hay que estar dándole a las teclas PKAY y STOP del cassette, por lo que va más rápido. También existe la posibilidad de escribir y leer repetidamente en el mismo archivo, cosa que en la cinta no se podía hacer, debido a que al escribir encima de un fichero ya existente, éste se destruía completamente y había que volverlo a escribir entero. En los discos se puede grabar sólo una parte, con lo que los típicos programas encargados de llevar archivos de clientes y similares funcionan mucho mejor.

Por último, existe un conjunto de utilidades en CP/M que amplían aún más las ya comentadas: formateado de discos, copia de los mismos, ensamblador, un *debuger* y otros comandos que nos dan información del sistema.

El Logo

Una sorpresa agradable que se recibe al comprar el sistema es que en el precio se incluye el lenguaje Logo. Aunque muchas cosas ofrecen programas como complemento de los discos para incentivar las ventas, ésta es la primera vez que vemos que se proporciona un lenguaje de programación; además, éste ya posee una gran reputación por sus buenas cualidades para la enseñanza. En especial, el que se incluye es **Dr. Logo**, la versión realizada por **Digital Research** y una de las más extendidas, ya que es la misma que usa el **IBM PC** y otros similares.

Respuesta del sistema

Para medir la rapidez de los discos cogimos dos máquinas, una con discos y otra con cinta, y empezamos a medir tiempos en operaciones similares y que se realizan habitualmente, como la carga de un programa, la grabación, etc. El resultado, como era de esperar, fue realmente impresionante, ya que aunque la velocidad de escritura del Amstrad en cinta es elevada, los discos demostraron estar muy por encima. Además se debe recordar que en este caso no hace falta pulsar la tecla PLAY y luego otra del teclado como ocurría con la cinta. Los tiempos comparativos que obtuvimos de carga de un programa fueron de 16 segundos en disco y de 5 minutos, 39 segundos en cinta. ¡Veinte veces más rápido! En grabación las diferencias fueron similares, dando 11 segundos en disco y 3 minutos, 50 segundos en cinta.

Hay que indicar que pueden surgir problemas en algunos casos especiales, debido a que algunos programas que vienen en cinta consumen mucha memoria y «chocan» con la *ROM* del sistema operativo. Pero esto se puede arreglar fácilmente con una pequeña rutina que damos en la sección de trucos de este mismo número.

Conclusión

La superioridad de los discos ha quedado claramente demostrada y puede que muchos lectores hayan visto ya que sí les interesa. De todos modos no son la panacea universal y en algún caso pueden resultar superfluos. Por ejemplo, aquellos usuarios que se dediquen casi exclusivamente a usar juegos comerciales le encontrarán poca utilidad, ya que éstos suelen ir protegidos y no permiten el ser pasados a disco. No obstante, si usted quiere utilizar programas más serios y ve el ordenador como un ayudante más que como una máquina de juegos, entonces indudablemente le hace falta, ya que todos los programas de utilidad (procesadores de textos, contabilidades, etc.) suelen ir en disco, resultando mucho más efectivos que si van en cinta, debido a las posibilidades que ofrecen.











Programa: Grand Prix Driver
Tipo: juego
Distribuidor: Indescomp
Formato: cassette



Ningún ordenador personal que se precie puede dejar de tener carrera de coches con la que podamos demostrar nuestra pericia automovilística (que no tiene nada que ver con la capacidad real de conducir). Este es uno de los programas en el que tenemos que correr un gran premio adelantando a todos los coches que se nos pongan por delante.

La carga del juego se realiza del modo habitual y una vez terminada nos aparece la pantalla de presentación donde se nos muestran las teclas que podemos usar o el *joystick* si tenemos uno. Existen cuatro controles principales: dos para desplazarse a izquierda y derecha, uno para acelerar y otro para frenar. El objetivo del juego es adelantar a los treinta coches que van por delante de nosotros sin chocar ni salirnos de la pista hasta lograr colocarnos en primera posición. Una vez se empieza, se dibuja un paisaje al fondo junto con la carretera que es vista en perspectiva como es habitual en este tipo de juegos y los coches, que para dar la misma sensación, son mostrados por detrás.

La conducción resulta algo difícil debido a que los controles, de un modo similar a la vida real, tardan algo en responder y no se dispone de la agilidad habitual en otro tipo de juegos. Esto, unido a que el tamaño de la carretera no es muy grande, hace que el juego sea bastante difícil de dominar y los choques son muy habituales provocando la destrucción instantánea de nuestro vehículo.

Aparte de esto, en la parte inferior de la pantalla existen unos indicadores que, entre otras cosas marcan la velocidad que llevamos, velocidad que resulta bastante más real que en otros juegos (a 70 km/h ya resulta bastante difícil adelantar).

El juego en definitiva carece de la velocidad y animación de otros, debido sobre todo a que emula bastante bien a los reales. Aún así resulta complicado de dominar y puede representar un buen reto para jugadores expertos.



ADICCION: 5
PRESENTACION: 6
GRAFICOS: 5
ACCION: 6

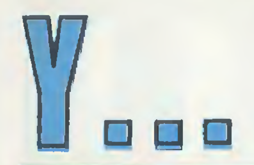
Programa: Alien
Tipo: juego
Distribuidor: ABC Soft
Formato: cassette



Nos encontramos ante otro programa de la larga lista de «importados» del Spectrum. En este caso se trata de un popular juego de la casa ULTIMATE, que se ha ganado una merecida reputación por la calidad de sus programas.

El escenario de nuestra aventura es una nave espacial en la que se halla un robot (nuestro protagonista) que debe realizar ciertos actos para salvar la nave y su tripulación. Tripulación que se halla criogenizada en cápsulas de las que tenemos que conservar funcionando. Evidentemente esto no es sencillo y existen multitud de peligros que deberemos evitar. Además de esta misión hay que ir haciendo otra serie de tareas que permiten que la nave siga en funcionamiento.

Al cargar el programa nos aparece una pantalla de presentación realmente bonita y después de algunos minutos aparece el juego en sí. Este está hecho con una técnica tridimensional bastante usada por ULTIMATE y que le da una presentación y animación realmente sorprendente, ya que los objetos pasan unos delante de otros con un realismo fuera de lo común, aunque en algunas situaciones se produzcan errores de apreciación sobre la situación real de nuestro per-



sonaje, lo que ocasiona que se cometan errores, a veces fatales, para el desarrollo del juego. El manejo también es algo complicado, ya que al contrario de la mayoría que tiene cuatro teclas para indicar las cuatro posiciones, aquí hay dos teclas que hacen que gire a un lado o a otro, otra que lo hace avanzar y otra para saltar. Esto enlentece el juego debido al poco control. No obstante, no es tan importante como parece ya que el juego no es de mucha rapidez, sino más bien de habilidad e ingenio.

La presentación del juego es asombrosa, con unos gráficos y una acción que servirán de estándar durante mucho tiempo. Pese a estar basado en acciones, en lugar de conversaciones, se puede encuadrar dentro de los juegos de aventura, por lo que puede servir para esas largas noches de invierno.



ADICCION: 7
PRESENTACION: 8
RAPIDEZ: 9
CLARIDAD: 6

Programa: Manic Miner
Tipo: juego
Distribuidor: Indescomp
Formato: cassette



El juego sigue el conocido esquema de varios niveles de juego a los que se va pasando un vez se logra superar el anterior. Cada nivel es una habitación en la que se deben recoger todas las llaves necesarias para abrir la puerta que nos da el paso a la siguiente. Todas ellas disponen de diversos peligros que hay que evitar por medio de las tres teclas del juego. Dos de la fila superior de letras controlan el movimiento a la izquierda y derecha y una de la inferior hace que **Willy** (el minero que controlamos) salte. Este salto es vertical si estamos parados o a izquierda o derecha si nos estábamos moviendo en esa dirección.

Los peligros que tenemos que evitar son de muy variada clase y forma. Los hay fijos, como son las arañas y salientes del techo y los móviles, que son autómatas abandonados.

Aparte de hacer que nos caigamos si nos quedamos quietos encima de ellos, aportan la propiedad al juego de que en muchas pantallas es necesario hacer el recorrido exacto a la primera, porque sino, al intentar repetirlo ha desaparecido el suelo.

Unido a ésto se halla una cantidad limitada de aire que va disminuyendo a medida que permanecemos en una pantalla. Aparte de la necesidad de conservar este aire porque si se acaba, termina el juego.

El juego resulta fácil de manejar, y después de algunos minutos de juego se logra pasar de la primera pantalla, el problema es que como hay bastantes, resulta muy difícil hacerlas todas bien y al menor descuido hay que volver a empezar. También se debe tener en cuenta muchas situaciones hay que realizar saltos calculados al milímetro.

La realización del juego está muy cuidada en gráficos y sonido.



ADICCION: 9
PRESENTACION: 7
GRAFICOS: 6
ACCION: 7





Programa: Star comando
Tipo: juego
Distribuidor: Indescomp
Formato: cassette



Este juego de evidentes connotaciones galácticas es una recreación de uno de los primeros juegos de ordenador, *Star Trek*. En aquella versión había que recorrerse todo el universo, compuesto de dieciseis sectores, y destruir a todos los *Klingons*. Como su creación fue conjunta a la de los primeros ordenadores, no poseía características que hoy se consideran fundamentales: gráficos, sonido, acción, etc. sino que era un juego parecido a los barquitos.

Esa misma idea de limpiar la galaxia de peligros ha sido retomada en este juego y actualizada añadiéndose todos esos ingredientes que hacen de él un juego realmente interesante. La galaxia sigue dividida en 16 sectores y debemos explorarlos y destruir todos los peligros que haya en cada uno de ellos. En cada sector el juego nos presenta la galaxia vista desde la cabina del piloto con el punto de mira en el centro. Por medio del *joystick* (imprescindible para este juego) podemos desplazarnos a un lado u otro y así apuntar a los objetos que se lanzan sobre nosotros. Estos objetos pueden ser distintos en cada sector y hay meteoros, piratas galácticos y todo un muestrario de peligros estelares. El botón del *joystick* hace que se disparen ráfagas de tres disparos y si lo mantenemos pulsado durante cierto tiempo nos muestra un menú donde además de indicarnos la energía que nos queda y los puntos conseguidos, podemos elegir a que sector de los 16 existentes podemos trasladarnos. Para ello se utiliza también el *joystick* que hace que una flecha que indica el sector se desplaze hacia arriba o abajo. Una vez elegido un sector pulsamos el botón y el ordenador nos envía a él, para seguir jugando. En dicho menú también se da información de lo que hay en cada sector, indicando si está vacío. Pero en muchos casos la información es «datos no disponibles», con lo cuál hay que arriesgarse para averiguarlo.

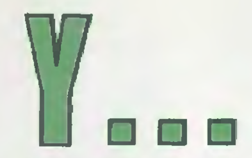
Debido a que la energía disponible se gasta con rapidez existen naves nodriza en las que podemos recargar. Para ello nos trasladamos al sector correspondiente y al aparecer hay que hacer coincidir la mirilla con un diamante situado en su parte central, al hacer esto y pulsar el botón del *joystick* automáticamente se recargará la energía. Cuanto más tiempo lo tengamos centrado más energía recibiremos.

Una opción interesante que se nos ofrece antes de empezar el juego es la de si queremos que el control sea de tipo piloto o arcade. En ambos casos la presentación del juego es la misma, pero en el último el mando responde como es de esperar instintivamente (al bajar, los enemigos bajan y al subir, los enemigos suben) mientras que en la opción de piloto, la acción se desarrolla como si fuesen los mandos de un avión normal, haciendo que los enemigos suban cuando bajamos el mando y viceversa, que lo hace más realista pero también más difícil de manejar.

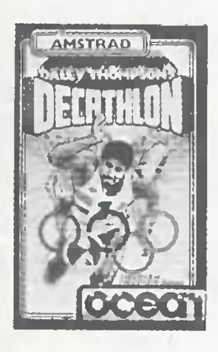
El juego está bien realizado y posee todos los ingredientes necesarios para ser uno de los mejores de acción. Los gráficos están bien hechos y la sensación de perspectiva conseguida al aumentar o al disminuir de tamaño las naves es muy buena. Aunque quizás no tan adictivo como otros juegos, sin duda proporcionará buena diversión a los jugadores.



ADICCION: 6
PRESENTACION: 6
GRAFICOS: 7
ACCION: 8



Programa: Decathlon
Tipo: Juego
Distribuidor: Erbe
Formato: cassette
Precio: 2.100 Ptas



Una de las tendencias más importantes de las casas de **software** en los últimos tiempos consiste en escribir los programas pensando ya en su conversión a otros ordenadores. Cuando apareció el Amstrad, muchas compañías «convirtieron» sus programas para que pudieran funcionar en la nueva máquina, a veces con resultados muy espectaculares. El **Decathlon** es un ejemplo de esta tendencia.

El reto que nos plantea este juego es importante. Se trata de participar en la prueba más completa del calendario olímpico: para tener éxito hace falta velocidad (100 y 400 metros longitud), buena coordinación (salto con pértiga, altura y 110 vallas), y potencia (peso, jabalina y disco). Para que no falte nada, hay que resistir la prueba de 1.500. Agotador, como se puede ver.

El mecanismo es el mismo en todas las pruebas: para acelerar se mueve la palanca con rapidez de izquierda a derecha, y los saltos se realizan presionando el botón de disparo (Pobre **joystick**).

El sistema de movimiento es correcto en las pruebas de velocidad, aunque es necesario mantener el joystick en movimiento durante los saltos y lanzamientos. Asimismo, la prueba de 1500 exige una consideable dosis de esfuerzo a nuestro brazo y a la palanca; un sistema en el que los movimientos variaran en este tipo de pruebas haría más agradable el juego con este programa.

La representación gráfica es mucho mejor que la de otras versiones del mismo juego, con un escenario lleno de colores. En cambio, la animación del personaje no es todo lo buena que sería deseable, contando con las posibilidades de la máquina. Hay que tener en cuenta, sin embargo, que en la versión CPC del programa no hace falta cargar las pruebas en dos partes, como ocurre en otras máquinas.

Un programa con mucha acción, donde resulta casi inevitable entablar competiciones con los amigos. Las agujetas en los brazos pasarán en pocos días.

Hay, para finalizar, dos puntos a tener en cuenta antes de correr a la tienda a buscar una copia: el programa no es compatible con el CPC-664, por los que los usuarios de disco tendrán que esperar una mejor ocasión. Los poseedores del CPC-464 también deben tener en cuenta que el joystick es absolutamente necesario para jugar. Con estas salvedades, un juego realmente excelente.



ADICCION: 8
PRESENTACION: 8
GRAFICOS: 9
ACCION: 6

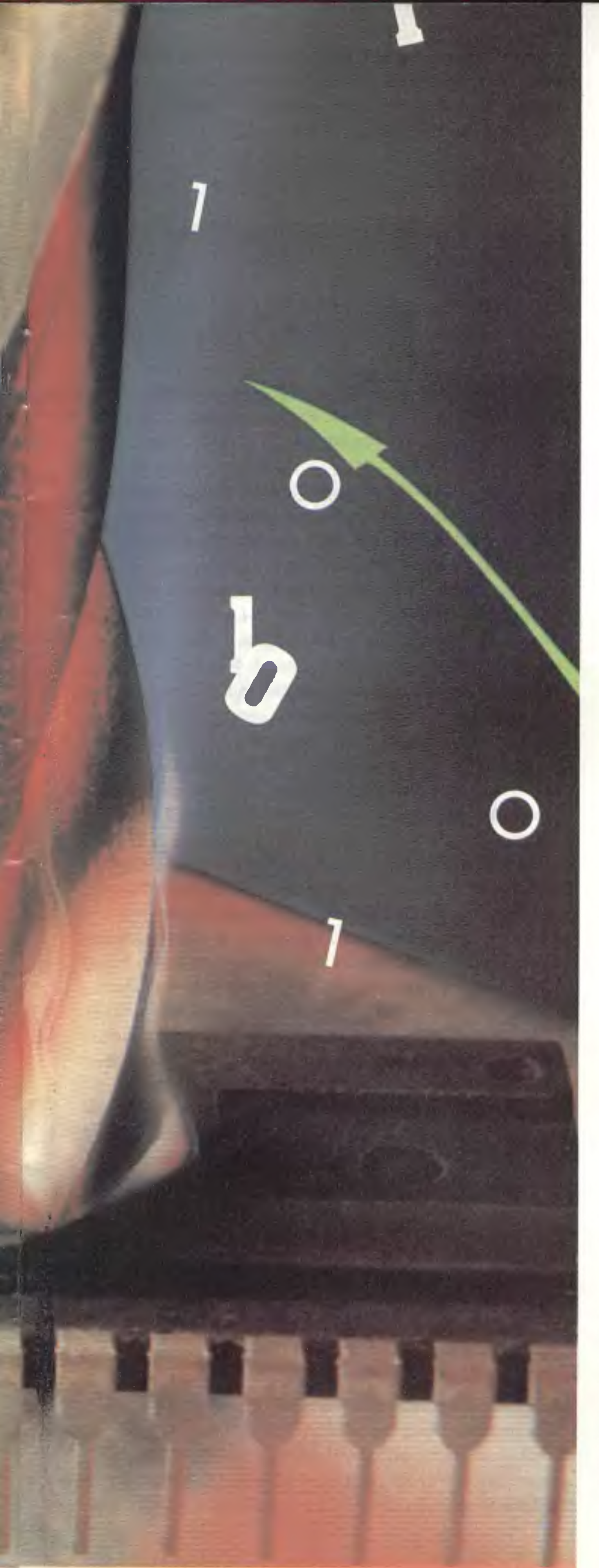
COMENTADO

LA LOGICA BINARIA ES LA BASE



Aunque el título de este artículo puede asustarle haciéndole pensar que trata de temas extraordinariamente complejos, la realidad es distinta. Pretendemos explicar algunos términos de uso común pero que mucha gente no entiende; asimismo, veremos qué es una operación lógica y cómo podemos usarla desde BASIC.

Para empezar hay que saber que el ordenador tiene un conjunto muy básico de dispositivos que son los que le hacen funcionar: La Unidad Central de Procesos (CPU), la memoria RAM, la memoria ROM y los dispositivos de entrada y salida. Cada uno de ellos tiene funciones muy específicas que explicaremos brevemente.



A FONDO

La Unidad Central de Procesos es el corazón de la máquina, ya que es la encargada de ejecutar todas las instrucciones que la damos. Para poder hacer esto dispone de un catálogo de operaciones que puede ejecutar. Las operaciones son muy sencillas (por ejemplo, no sabe multiplicar) y si hay que hacer algo más complicado se le tiene que explicar como se hace en función de esas instrucciones. El lenguaje que entiende es el código máquina del que se habla algo más en otro artículo de este número. La memoria *RAM* es una memoria donde se pueden almacenar datos que nos interesen para usarlos posteriormente. Los contenidos de esta memoria se pueden modificar siempre que queramos y se borrar al apagar el ordenador, por lo que si no lo hemos salvado en cinta o en disco, los perderemos. La otra memoria *ROM* es sólo de lectura, es decir, podemos leer lo que hay grabado en ella, pero no podemos modificarlo, y sus contenidos no se borran aunque se vaya la luz. En la *ROM* del Amstrad se encuentra el intérprete de *BASIC*. Este es un programa que se encarga de traducir las instrucciones que le damos en *BASIC* a código máquina para que pueda extenderlas la *CPU*. Por último los dispositivos de entrada y salida son los que se encargan de comunicar al ordenador con el mundo exterior (es decir, usted), hay uno encargado de generar la imagen que vemos por la pantalla, otro graba en cinta, otro genera sonidos, etc.

El ordenador considera, en lenguaje máquina, que la memoria es un conjunto de casillas numeradas consecutivamente y que en cada una de ellas se puede almacenar un *byte* (término que explicaremos posteriormente). Naturalmente, desde *BASIC* esto no lo notamos, ya que el intérprete se encarga automáticamente de usar la memoria que le haga falta sin que nosotros le tengamos que deoir cual. No obstante, podemos manejarlas directamente desde BASIC, si queremos, por medio de las instrucciones *PEEK* y *POKE*. Si ponemos *PRINT POKE* (número) en el Amstrad (donde el número está comprendido entre el 0 y 65535 que son las memorias que tiene el ordenador), nos mostrará por pantalla el contenido de dicha memoria, y si hacemos *POKE* dirección, número, guardará el número (entre 0 y 255, ya que este es el rango que cabe en un *byte*) en la memoria que tenga la dirección indicada. Tenga cuidado al pasar esta última instrucción ya que es muy fácil dejar tonto al ordenador (pero todo se arregla apagando y encendiendo la máquina). Naturalmente, si intentamos guardar algo en la memoria *ROM*, el ordenador ignorará la orden.

Bits y bytes

Estas dos palabras mágicas que se pronuncian mucho en el mundo de la informática y muchas veces sin saber su significado exacto. La gente, incluyendo a muchos vendedores, dicen que tal ordenador tiene una memoria de 64 kilobytes sin saber que indica eso. Para resolver el entuerto vamos a dar una breve explicación de cada uno de los dos términos, por razones de cultura general y también porque será útil después, cuando veamos como usar algunas instrucciones del *BASIC* que tienen una relación con esto (aparte de los *PEEK* y *POKE* ya comentados).

En primer lugar veamos el bite. El nombre proviene del término inglés *Binary digit*, que traducido literalmente significa dígito binario. Si tiene unos conocimientos matemáticos básicos, sabrá que un dígito es cada una de las cifras que componen un número, en este caso el número en cuestión es binario, lo que quiere decir que trabaja en base dos. Normalmente nosotros trabajamos en base 10, que indica que con una sola cifra se pueden expresar diez cantidades distintas (0,1,2,3,4,5,6,7,8 y 9). En cambio, en binario, con una sola cifra sólo se pueden expresar dos cantidades distintas: 0 y 1. Esto hace que el sistema de numeración cambie a uno distinto del que conocemos y usamos. Un bit, para verlo de un modo más claro, puede representarse como un interruptor que enciende o apaga una bombilla. Si la bombilla está encendida, decimos que tiene el valor uno y si está apagada, entonces tiene un valor 0. Esta es la mínima información que se puede almacenar y usar en cualquier sistema y los ordenadores almacenan todos los datos que se le dan por medio de bits, combinándolos para poder guardar más información. Sin embargo, y para evitar dolores de cabeza al usuario, al construir un ordenador, el fabricante introduce unos programas especiales que transforman del sistema binario al decimal y viceversa, de modo que haga la conversión de uno a otro automáticamente y no tengamos que hacerla nosotros.

El método más común de agrupar los *bits* es en *bytes*. Un *byte* es un conjunto ordenado de 8 *bits*. Es importante tener en cuenta que decimos ordenado, ya que significa que no todos los *bits* valen lo mismo. Igual que si trabajando en décimal un 7 no vale lo mismo si está a la derecha o a la izquierda de otra cifra, con los *bits* pasa lo mismo y no valen lo mismo si están en la primera posición o en la última.

Los *bytes* son tan importantes porque suelen constituir la unidad principal con la que trabaja el microprocesador (en especial el Z80 que lleva el Amstrad) que es capaz de leer o escribir de una vez. Cuando en las operaciones intervienen números más complejos, lo hace enlazando varios bytes, leyéndolos y escribiéndolos en secuencia. Con ellos se pueden representar números, caracteres e, incluso, instrucciones del programa, ya que el ordenador les asigna una función u otra según le digamos.

Pasando de decimal a binario y viceversa

Hemos comentado que un número binario se puede pasar a decimal y viceversa. El proceso necesario para hacer esta conversión es sencillo una vez que se aprende a realizarlo, como vamos a ver. Hemos dicho que los bits de un byte tienen un valor distinto según la posición en que se encuentren. Esta colocación se hace, al igual que en decimal, empezando desde la derecha, pero el *bit* situado más a la derecha se denomina el cero en vez del uno y el de más a la izquierda es el siete en vez del ocho. La razón de utilizar esta notación es que simplifica los cálculos cuando queremos pasar el número a decimal.

Si imaginamos un número en decimal, como puede ser el 538, asumimos que significa una cantidad determinada en función de la posición que ocupa cada cifra. El ocho, al estar a la derecha, no varia (o por decirlo de otro modo, se multiplica por uno), pero en cambio el tres indica 3 por 10, y el cinco es 5 por 100. Si la posición de cada número la numeramos de derecha a izquierda empezando desde el cero, como hemos hecho en binario, veremos que 10 elevado a cero (la primera posición) da 1, que por lo que habíamos multiplicado el 8, 10 elevado a uno (la segunda posición) da 10, que es lo que habíamos multiplicado al tres y 100 elevado a dos nos da 100, que multiplica a cinco.

Este mismo proceso, que en decimal se da por supuesto, hay que realizarlo en binario para convertir el número a decimal, pero teniendo en cuenta que, al ser base dos, no hay que elevar diez a una potencia, sino 2. Y la posición de la derecha será 2 elevado a 0 (1), la siguiente 2 elevado a 1 (2), la siguiente 2 elevado a 2 (4) y así sucesivamente hasta llegar a 2 elevado a 7 (128) que es el situado en la posición de más a la izquierda del byte. Si operásemos con números de más extensión (por ejemplo de 16 bits) el proceso seguiría elevando 2 a los sucesivos números de orden, 8, 9...15. El proceso de conversión a decimal consiste, entonces, en elevar al orden del *bit* y multiplicarlo por el valor de éste. Por ejemplo, el número binario 00010101 será 1*1 de la posición cero maps 1*4 (2 elevado a 2) de la segunda posición (recuerde que se cuenta desde cero) más 1*16 (2 elevado a 4) que da 21. El resto de los *bits* da un cero (el valor) multiplicado por otro número, con lo que se obtiene cero y por tanto se desprecian.

```
1 MODE 1:PAPER 0:CLS 'MANEJO DE BITS
10 WINDOW #1,8,35,3,3
20 WINDOW #2,8,34,5,5
30 WINDOW #3,8,34,7,7
40 WINDOW #4,8,34,8,8
50 LOCATE #1,1,1
60 CLS#1:INPUT#1,"INTRODUZCA SU NUMERO";A
70 LOCATE #2,1,1
80 PEN#2,2:INK 2,5:CLS#2:PRINT#2,"EL NUMERO = ";A
90 LOCATE#3,1,1:PEN#3,5:INK 5,24
100 PRINT#3,"BIT  0 1 2 3 4 5 6 7 "
110 LOCATE#4,6,1:PEN#4,6
120 FOR I=0 TO 7
130 IF A AND 2^I THEN PRINT#4,"1 ";ELSE PRINT#4,"0 ";
140 NEXT
150 GOTO 50
```

El proceso inverso es algo más laborioso, debido a que implica unas cuantas restas y comparaciones como se explica a continuación. Si nos dan un número en decimal y queremos pasarlo a binario, como por ejemplo el 152, en primer lugar hay que partir determinando el número de bits que vamos a usar. En muchos casos este dato nos viene dado, por ejemplo, si queremos convertir el número obtenido por medio de un *PEEK*, sabemos que es un *byte*. Si no tenemos ese dato, lo determinaremos a ojo de modo que quepa (si el número de bits no fuese lo suficientemente grande nos saldría un error inmediatamente, como veremos después). En nuestro ejemplo consideraremos que es un *byte*. Una vez hecho esto vamos cogiendo todas las potencias de dos, empezando por la del *bit* más alto que vayamos a usar y las vamos comparando con el número en cuestión, si son mayores las potencias, el *bit* correspondiente se pone a cero y no se hace nada, si por el contrario son menores o iguales, ponemos el *bit* a uno y le restamos la potencia al número para quedarnos con el resultado. Después de eso pasamos a la potencia inmediatamente inferior, y así sucesivamente hasta llegar al 1 (2 elevado a cero). En nuestro caso primero probamos con 2 elevado a 7 (128) y como es menor, ponemos el *bit* 7 a 1 y restamos 128 a 152, quedando 24. A continuación probamos con 64, pero al ser mayor, ponemos el *bit* correspondiente a 0 y pasamos a la siguiente que es 32, al ser también mayor se vuelve a repetir el proceso y llegamos a 16 que es menor, por lo que el *bit* 4 se pone a uno y se resta, quedando 8. La potencia inmediatamente es, precisamente 8, por lo que el *bit* 3 también se pone a 1 y se resta quedando de resultado cero, lo que hace que evidentemente, todos los *bits* inferiores valgan cero, ya que las potencias de dos correspondientes serán siempre mayores que cero. El número binario obtenido es, pues, 10011000 que se puede comprobar pasándolo de nuevo a decimal por el proceso descrito anteriormente. Después de la conversión el resto tiene que ser siempre cero, como nos queda a nosotros en el ejemplo después de restar cero, porque sino se habría cometido un error. Cuando restemos una potencia al número que se está convirtiendo, el resultado siempre tiene que ser menor que la potencia, si fuese mayor indicaría que usamos pocos *bits* y que hay que usar más. Esto siempre se detecta en la primera comparación. En nuestro caso está bien, ya que al restar 128 a 152 queda 24 que es menor que 128, pero si intentamos convertir 271, por ejemplo, también con ocho *bits*, al restarle 128 queda 143 que es mayor que 128 y, por tanto, está mal. Otro tanto sucede con 256, al restarle 128 queda 128 que es igual a la potencia, esto indica que hace falta usar un *bit* más, ya que la potencia de dos de ese *bit* será precisamente 256.

Operaciones lógicas

La razón de que estas funciones no sean más conocidas, muchas veces no es culpa del programador, sino de la pobre implementación de las funciones booleanas (otro nombre de las funciones lógicas) en la *ROM* de la máquina. Con el CPC 464 y el 664, sin embargo, no hay excusa posible. El *BASIC* del Amstrad es muy potente y permite realizar todo el conjunto de operaciones lógicas desde sentencias del *BASIC*.

En el álgebra booleana sólo existen dos posibles soluciones a una operación FALSO (0) o VERDAD (1) al igual que con los números binarios. Si el resultado de una operacion es FALSO, el ordenador pone todos los *bits* del *byte* correspondiente a cero y si es VERDAD, entonces pone los ocho a uno.

Cuando usted programa el ordenador con un IF, THEN y ELSE, está utilizando operaciones lógicas:

10 IF X=Y1 AND Y=2 THEN GOTO 30 ELSE GOTO 100

Si X=Y1 y Y=2 entonces la condición es cierta y el programa salta a la línea 30. Cualquier otro valor de X e Y hace que el programa salte a la línea 100. Expresiones lógicas del tipo IF G THEN GOTO 100 también se pueden usar en el Amstrad. Siempre que G sea distinto de cero (tenga algún *bit* igual a uno) el programa saltará a la línea 100. Esto es un truco muy útil que puede ahorrarle un montón de comparaciones.

El uso de *AND*, *OR* y *NOT* no está restringido a las sencillas relacciones que hemos mencionado. También se pueden usar para realizar operaciones booleanas, manipulación de bits, y comparaciones de bits específicos. Llegados a este punto no está de más ver cómo funciona cada operación con un número dado.

NOT

La expresión *NOT* hace el complemento a 2 del número mediante la inversión de todos los bits (los que

están a cero pasan a uno y los que están a uno pasan a cero) y la suma posterior de uno.

```
fórmula=>    NOT A=-(A+1)
NOT  1=>     resultado=-2
NOT -2=>     resultado=1
NOT  0=>     resultado=-1
```

AND

La operación *AND* se utiliza para poner a cero determinados *bits* y dejar los demas como estaban. En binario el *AND* lo que hace es que si los dos *bits* correspondientes están a uno, en cualquier otro caso (uno a cero y el otro a uno, o los dos a cero) el resultado es cero. Así:

```
12 AND  4=>    resultado= 4
25 AND 12=>    resultado=12
 4 AND  2=>    resultado= 0
```

Nota: Este último resultado se debe a que 4=0100 en binario y 2=0010. Cuando se hace el *bit* a *bit*, el resultado es cero.

OR

Esta operación también funciona con dos números como la anterior, pero en este caso al operar *bit* a *bit*, el resultado vale uno siempre que alguno de los *bits* (o los dos) valgan uno, y solo será cero si los dos *bits* correspondientes son cero.

```
 4 OR   2=>  resultado= 6
-1 OR  -2=>  resultado=-1
             11111111=-1
             11111110=-2
al hacer el OR  11111111=-1
```

Los problemas de convertir de decimal a binario se pueden resolver, sino se quiere trabajar mucho, con el listado de la figura 1, que además utiliza algunas d e l as funciones que acabamos de ver.

Pero, ¿qué significa todo ésto? Usando operaciones lógicas podemos poner a uno, a cero, y chequear *bits* directamente desde *BASIC*. Los programadores en lenguaje máquina apreciarán el significado de la última frase. Los programas funcionarán mas rápido, el espacio que ocupa el programa será más pequeño y se pueden meter ocho banderas «si/no» en un byte.

Las siguientes fórmulas le permitirán comprobar cualquier dato de un byte desde BASIC. Por ejemplo la función *JOY(0)* devuelve un número entero que depende del movimiento que haya realizado el jugador con el *joystick*. Si se ha movido a la derecha, el valor será un cuatro. Si se pulsa el botón de disparo a la vez, el valor devuelto será 20. Es, por lo tanto, muy sencillo escribir una subrutina que compruebe el estado de cada *bit* con las instrucciones dadas anteriormente.

Fíjese que el CPC 464 y el 664 almacenan los enteros como dos *bytes* contiguos, lo que permite que se almacenen números comprendidos en el rango −32767 a 32767. Las siguientes rutinas solo funcionan con enteros. Debe asegurarse de que las variables que use sean de este tipo, usando un *DEFINT* al principio del programa o usando el formato A%.

Fórmulas

– Para poner a uno un *bit* X en un entero de dos *bytes* NM.

NM=NM OR 2 X

para el *bit* 9: NM=NM OR 2 X

– Para mirar el valor de un *bit* X en un entero de dos *bytes* NM.

IF NM AND 2 X THEN ...ELSE...

para el *bit* 4: IF NM AND 2 4 THEN GOTO 1200 ELSE GOTO 50

– Para poner a cero un *bit* en un entero de dos *bytes* NM.

NM=NM AND NOT 2 X

para el *bit* 3: NM=NM AND NOT 2 3

También puede usar con caracteres alfanuméricos. CHR$(n) es un carácter de un *byte*. Creando una tira de la longitud que se quiera, se pueden tener tantas banderas como se quiera.

– Para poner a uno el *bit* X en un carácter T$ (tira de carácter).

T$=CHR$(ASC(T$) OR 2 X)

– Para poner a cero el *bit* X en un carácter T$.

T$= CHR$(ASC(T$) AND NOT 2 X)

– Para mirar un *bit* X en un carácter T$.

IF ASC(T$) AND 2 X THEN ...ELSE...

Para almacenar más de ocho condiciones lógicas, debe calcular cuantas condiciones necesita y crear una tira alfanumérica con la longitud apropiada a ese valor en función de la siguiente fórmula.

longitud(n)= INT(número de condiciones/8)+1

Entonces se puede crear la tira con

T$ = STRING$(n,0)

Que creará la tira deseada y pondrá cada *bit* a la condición inicial de cero.

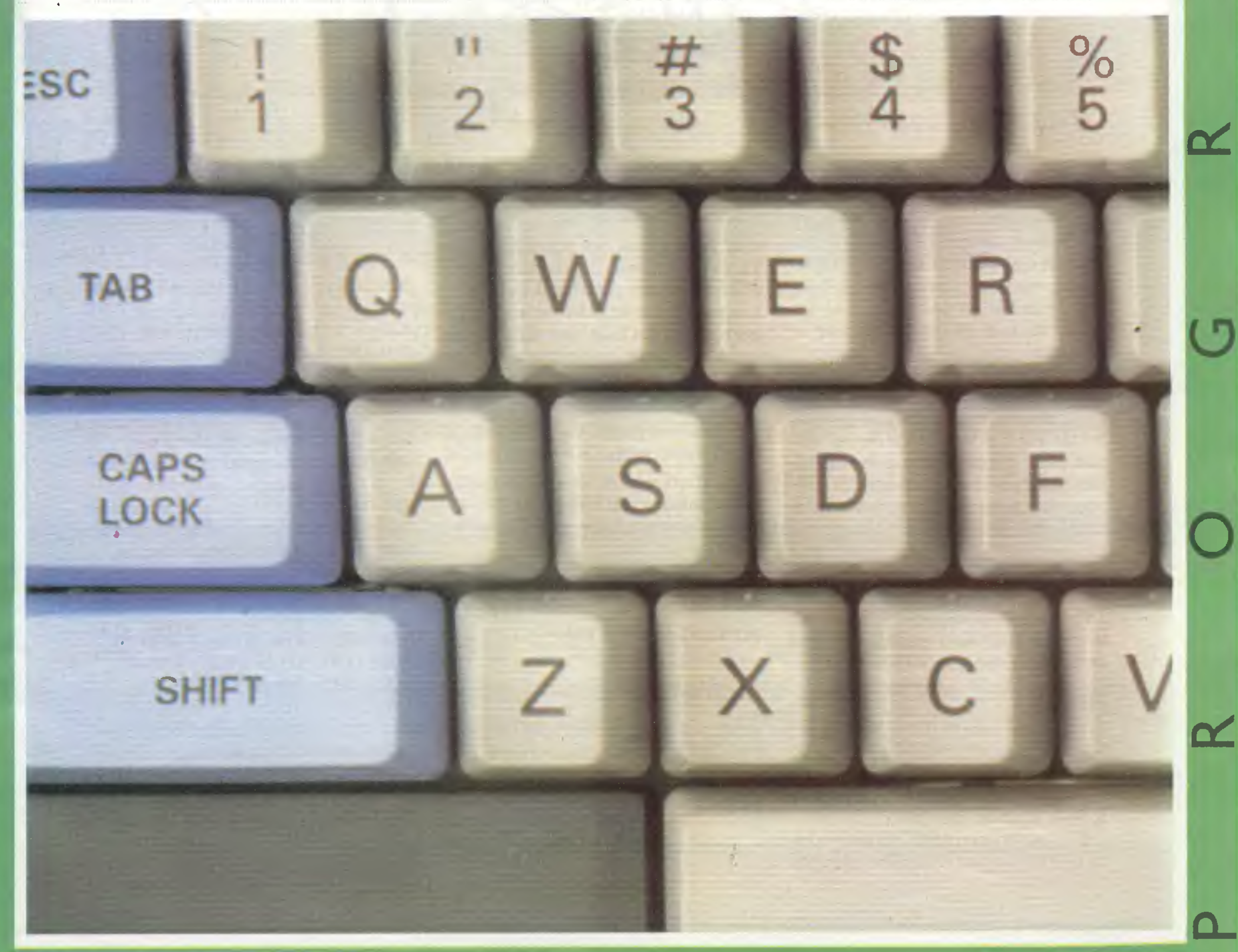
Todas las fórmulas anteriores se pueden mejorar definiéndolas como funciones por medio de DEF FN. Después de cavilar un poco encontrará muchos usos para las rutinas anteriores. Finalmente considere que:

```
X = X+2 : IF X> 3046 THEN X=3046
```

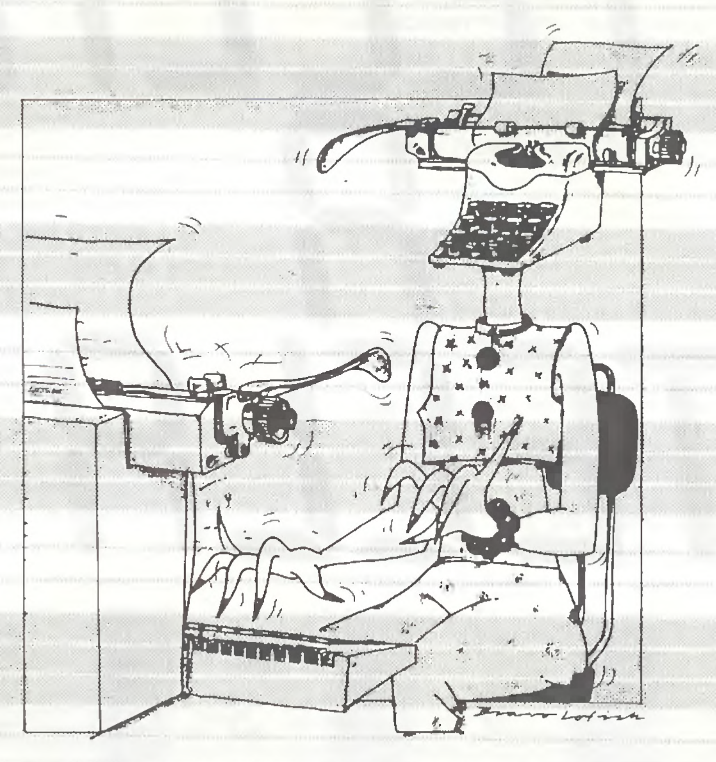
ES LO MISMO QUE

```
X=X-(X+2<3046)
```

TECLA A TECLA



MAQUINA DE ESCRIBIR



Este programa convierte al CPC 464/664 en un sencillo sistema de escritura del tipo de máquina de escribir. Tiene órdenes para salvar y leer el texto de disco o cinta, cortar e insertar líneas, modificar líneas ya introducidas y mostrar en pantalla con el formato «lo que se ve es lo que se imprime».

Es fácil de usar con una línea de menú en la parte superior. También tiene una utilidad de «pánico» que permite el ir al menú principal sin perder ningún dato. Simplemente pulse «ESC» dos veces y entonces la tecla «ENTER» pequeña (la del teclado numérico) para volver al menú.

El programa no es un editor de toda la pantalla pero tiene las mismas facilidades que el editor de líneas del BASIC. Cada línea está numerada y las funciones del menú manejan el texto de acuerdo con este número.

Notas

1. No use dobles comillas (") cuando vaya a guardar el documento en disco o cinta. El ordenador interpreta esto como delimitadores y terminará la línea cuando aparezca.

2. Cuando guarde los textos en disco asegúrese que el nombre es válido para el AMSDOS y el CP/M, es decir, ocho letras para el nombre y tres para el tipo: ABCDEFGH.DOC

3. Para borrar un documento que ya se ha impreso o guardado, pulse ESC dos veces y vuelva a ejecutar el programa con RUN. Si quiere volver pero sin perder lo almacenado en memoria pulse la tecla ENTER pequeña.

4. Durante la introducción de texto, si se pulsa F al principio de una línea sin nada más y luego ENTER, hará que el programa vuelva al menú principal.

El programa funciona con todas las impresoras compatibles con Centronics. Si está usando un monitor verde, entonces la línea

```
60 MODE 2:BORDER 0: INK 0,0: INK 1,26: PAPER 0: PEN: 1: GOSUB 1370
```

producirá la mejor visión.

Comentarios

4050	Hace que todas las variaciones sean enteras. Redefine las teclas. Hace un redireccionamiento de los errores para manejar las órdenes no válidas.
60	Pone los colores, modos y ventanas.
70	Crea un área de almacenamiento para el disco (o cinta) en memoria.
80-120	Permite leer un fichero externo.
130	Manejo de errores.
140-180	Lee un fichero externo.
190-210	Pone las condiciones para la entrada de textos.
220-280	Entrada del texto.
300-460	Impresión del texto.
480-600	Muestra el texto por pantalla.
620-690	Modifica una línea de texto ya introducida.
710-780	Añade el texto ya existente.
800-870	Guarda el texto en cinta o disco.
890-1080	Inserta líneas de texto.
1100-1150	Corta líneas de texto.
1170-1320	Menú.
1390-1470	Subrutina que muestra una línea de texto.
1490-1610	Subrutina que permite teclear una línea e indica las que quedan libres.
1630-1690	Acepta el punto inicial y final tecleados por el usuario.

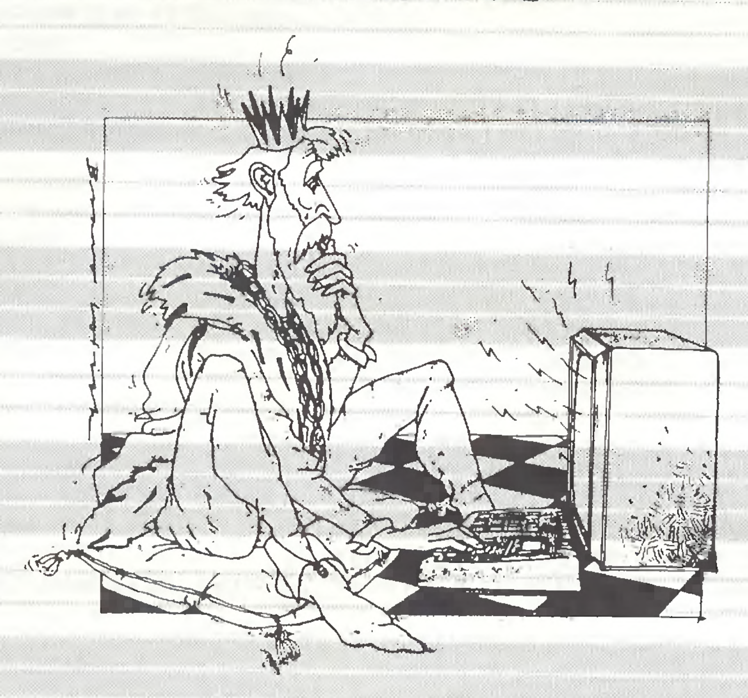
```
10 ' Maquina de Escribir
20 ' AMSTRAD USER 1985
30 '
40 DEFINT F,I,P,S,X: KEY DEF 18,1:KEY 139,"CLG:GOTO 1170"+CHR$(13)
50 REM ON ERROR GOTO 130:KEY 140,"          ":KEY DEF 68,1,140
60 MODE 2:BORDER 26:INK 0,26:INK 1,13:PAPER 0:PEN 1:GOSUB 1340
70 OPENOUT "FILEBUF":MEMORY HIMEM-1:CLOSEOUT
80 SOUND 1,20:PRINT#4,"QUIERE LEER UN ARCHIVO EXTERNO (S/N):"
90 F$=INKEY$:IF F$="" THEN 90
100 IF UPPER$(F$)="S" THEN 140
110 IF UPPER$(F$)="N" THEN 190
120 GOTO 90
130 SOUND 1,200:PRINT#4,"ENTRADA NO VALIDA!":FOR T=1 TO 2000:NEXT:RESU
ME
140 SOUND 1,20:INPUT#4,"INTRODUZCA EL NOMBRE DEL FICHERO";G$
150 SOUND 1,20:PRINT#4,"INSERTE EL DISCO O LA CINTA Y PULSE UNA TECLA.
"
160 IF INKEY$="" THEN 160
170 WINDOW SWAP 0,4:OPENIN G$:INPUT#9,L,PW:MXL=INT(70000/PW):DIM TEXT$
(MXL)
180 FOR X=1 TO L:INPUT#9,TEXT$(X):NEXT:CLOSEIN:WINDOW SWAP 0,4:GOTO 11
70
190 SOUND 1,20:INPUT#4,"ANCHO DE LA IMPRESORA (10-70)";PW
200 IF PW<10 OR PW>70 THEN 190
210 MXL=INT(70000/PW):DIM TEXT$(MXL):F=0:B=24:K=0:GOSUB 1340
220 GOSUB 270:FOR X=1 TO MXL:GOSUB 1490:IF F=1 THEN 1170
230 IF K=1 THEN 250
240 NEXT
250 SOUND 1,200:PRINT#4,"NO PUEDO SEGUIR. - DEMASIADAS LINEAS."
260 FOR T=1 TO 3000:NEXT:GOTO 1170
270 PRINT#4,"F=FIN DEL TEXTO"
280 PRINT#2,STRING$(23,60);:RETURN
290 '
300 ' IMPRESION
310 '
320 IF L=0 THEN 1310
330 SOUND 1,20:PRINT#4,"IMPRIMO TODO EL DOCUMENTO (S/N):"
340 H=0:B$=INKEY$:IF B$="" THEN 340
350 IF UPPER$(B$)="S" THEN START=1:FINISH=L:GOTO 390
360 IF UPPER$(B$)="N" THEN GOSUB 1630
370 IF H=1 THEN 390
380 GOTO 340
390 PRINT#4,"LA IMPRESORA NO ESTA CONECTADA":SOUND 1,200:PRINT#8,CHR$(
13);
400 SOUND 129,0:WIDTH PW:CLG:FOR X=START TO FINISH :PRINT#4,"LINEA";X
410 PRINT#8,TEXT$(X):GOSUB 1390:NEXT
420 SOUND 1,20:PRINT#4,"REPITO LA IMPRESION (S/N)?"
430 C$=INKEY$:IF C$="" THEN 430
440 IF UPPER$(C$)="S" THEN 390
450 IF UPPER$(C$)="N" THEN 1170
460 GOTO 430
470 '
480 ' PANTALLA
490 '
500 CLG:IF L=0 THEN 1310
510 SOUND 1,20:CLS#4:LOCATE#4,20,1:PRINT#4,1;SPC(3)"[ENTER]":LOCATE#4,
1,1
520 INPUT#4,"MUESTRO DESDE LA LINEA ";START:IF START=0 THEN START=1
530 IF START>L OR START<0 THEN 510
540 PRINT#4,"PULSAR ESPACIO PARA MOSTRAR"SPC(35)"T=PARA TERMINAR";
550 SOUND 1,20:FOR X=START TO L
```

MAQUINA DE ESCRIBIR

```
560 E$=INKEY$:IF E$="" THEN 560
570 IF E$=" " THEN 600
580 IF UPPER$(E$)="T" THEN 1170
590 GOTO 560
600 GOSUB 1390:NEXT:GOTO 1170
610 '
620 ' VOLVER A TECLEAR
630 '
640 IF L=0 THEN 1310
650 SOUND 1,20:INPUT#4,"INTRODUZCA EL NUMERO DE LINEA QUE VA A VOLVER
A TECLEAR";X
660 IF X<1 OR X>L THEN 650
670 PAPER#3,1:PEN#3,0:GOSUB 1390:LOCATE #3,1,24:LOCATE#0,1,24
680 PRINT#3,"TECLEE:>";:GOSUB 280:PAPER#3,0:PEN#3,1
690 LINE INPUT#0,"",TEXT$(X):CLG:GOSUB 1390:GOTO 1170
700 '
710 ' INTRODUCIR NUEVO TEXTO
720 '
730 IF L=0 THEN 1310
740 CLG:IF L>5 THEN S=(L-4) ELSE S=1
750 FOR X=S TO L:GOSUB 1390:NEXT:GOSUB 270
760 F=0:B=24:FOR X=(L+1) TO MXL:GOSUB 1490:IF F=1 THEN 1170
770 IF K=1 THEN 250
780 NEXT:GOTO 250
790 '
800 ' ALMACENAR
810 '
820 IF L=0 THEN 1310
830 SOUND 1,20:INPUT#4,"TECLEE EL NOMBRE DEL ARCHIVO";G$
840 SOUND 1,20:PRINT#4,"INSERTE EL DISCO O CINTA Y PULSE UNA    TECLA"
850 IF INKEY$="" THEN 850
860 WINDOW SWAP 0,4:OPENOUT G$:WRITE#9,L,PW:FOR X=1 TO L
870 WRITE#9,TEXT$(X):NEXT:CLOSEOUT:WINDOW SWAP 0,4:GOTO 1170
880 '
890 ' INSERTAR
900 '
910 IF L=0 THEN 1310
920 SOUND 1,20:INPUT#4,"INSERTAR DESPUES DE QUE LINEA ";START
930 IF START<0 OR START>L THEN 920
940 SOUND 1,20:INPUT#4,"CUANTAS LINEAS (MAX 50) ";I:IF I>50 THEN 940
950 FOR X=L TO START+1 STEP -1:TEXT$(X+I)=TEXT$(X):NEXT:L=L+I
960 FOR X=START+1 TO START+I :TEXT$(X)="":NEXT:CLG
970 PRINT#4,"LINEAS INSERTADAS ="I:FOR T=1 TO 1000:NEXT:Y=L
980 SOUND 1,20:PRINT#4,"QUIERE TECLEAR EL TEXTO (S/N):"
990 I$=INKEY$:IF I$="" THEN 990
1000 IF UPPER$(I$)="S" THEN 1030
1010 IF UPPER$(I$)="N" THEN 1170
1020 GOTO 990
1030 IF START>5 THEN V=(START-4) ELSE V=1
1040 FOR X=V TO START:GOSUB 1390:NEXT:GOSUB 270
1050 F=0:B=24:FOR X=START+1 TO START+I:GOSUB 1490:L=Y
1060 IF F=1 THEN TEXT$(X)="":GOTO 1170
1070 IF K=1 THEN 250
1080 NEXT:SOUND 1,20:PRINT#4,"INSERCION TERMINADA":FOR T=1 TO 2000:NEX
T:GOTO 1170
1090 '
1100 ' CORTAR
1110 '
1120 IF L=0 THEN 1310
1130 GOSUB 1650:PRINT#4,SPC(45)"TRABAJANDO - POR FAVOR ESPERE!"
1140 FOR N=1 TO(FINISH-START)+1:FOR X=START TO L:TEXT$(X)=TEXT$(X+1):N
```

```
EXT
1150 L=L-1:LOCATE#4,1,1:PRINT#4,"ORDENANDO LINEA ";X SPC(3):NEXT:CLG
1160 '
1170 ' MENU
1180 '
1190 GOSUB 1340:SOUND 1,20:PRINT#4,"ELIJA:>";
1200 PRINT#4,TAB(12)"M=MOSTRAR P=IMPRIMIR V=CORREGIR A=AGREGAR C=CORTA
R G=GUARDAR I=INSERTAR"
1220 A$=INKEY$:IF A$="" THEN 1220
1230 IF UPPER$(A$)="M" THEN 480
1240 IF UPPER$(A$)="P" THEN 300
1250 IF UPPER$(A$)="V" THEN 620
1260 IF UPPER$(A$)="A" THEN 710
1270 IF UPPER$(A$)="C" THEN 1100
1280 IF UPPER$(A$)="G" THEN 800
1290 IF UPPER$(A$)="I" THEN 890
1300 GOTO 1220
1310 SOUND 1,200:PRINT#4,"NO HAY NADA EN MEMORIA"
1320 FOR T=1 TO 2000:NEXT:GOTO 740
1330 '
1340 ' DEFINICION DE VENTANAS
1350 '
1360 WINDOW #0,9,(PW+8),3,25:WINDOW#2,(PW+9),(PW+9),3,25:WINDOW#3,1,8,
3,25
1370 WINDOW#4,1,79,1,1:PAPER#4,1:PEN#4,0:CLS#4:RETURN
1380 '
1390 ' RUTINA DE MOSTRAR EN PANTALLA
1400 '
1410 LOCATE#0,1,24:LOCATE#3,1,24
1420 PRINT#3,X;:PRINT#3,TAB(8)">";:LT=LEN(TEXT$(X))
1430 IF INT(LT/PW)=LT/PW AND LT>0 THEN PRINT #0,TEXT$(X);
1440 IF INT(LT/PW)<>LT/PW OR LT=0 THEN PRINT#0,TEXT$(X)
1450 IF INT(LT/PW)<>LT/PW AND LT>PW THEN FOR N=1 TO INT(LT/PW):PRINT#3
:NEXT
1460 IF INT(LT/PW)=LT/PW AND LT>PW THEN FOR N=1 TO (INT(LT/PW)-1):PRIN
T#3:NEXT
1470 RETURN
1480 '
1490 ' SUBRUTINA DE ENTRADA DE LINEAS
1500 '
1510 COUNT=INT((FRE(0)-1000)/PW)
1520 IF COUNT<1 THEN CLS#2:CLS#3:K=1:RETURN
1530 LOCATE #4,44,1:PRINT#4,"NUMERO DE LINEAS LIBRES="COUNT;SPC(3)
1540 LOCATE#3,1,24:LOCATE #0,1,B
1550 PRINT#3,X;:PRINT#3,TAB(8)">";:LINE INPUT#0,"",TEXT$(X)
1560 L=X:IF UPPER$(TEXT$(X))="F" THEN L=X-1:F=1:RETURN
1570 LT=LEN(TEXT$(X)):B=24
1580 IF INT(LT/PW)=LT/PW AND LT>0 THEN B=23
1590 IF INT(LT/PW)=LT/PW AND LT>PW THEN FOR N=1 TO (INT(LT/PW)-1):PRIN
T#3:NEXT
1600 IF INT(LT/PW)<>LT/PW AND LT>PW THEN FOR N=1 TO INT(LT/PW):PRINT#3
:NEXT
1610 RETURN
1620 '
1630 ' SUBRUTINA INICIO/FINAL
1640 '
1650 SOUND 1,20:INPUT#4,"COMIENZO EN LINEA ";START
1660 IF START<1 OR START>L THEN 1650
1670 SOUND 1,20:INPUT#4,"NUMERO DE LA ULTIMA LINEA ";FINISH
1680 IF FINISH<START OR FINISH>L THEN 1670
1690 H=1:RETURN
```

REINADO



Este programa proporciona el primer juego del tipo de «simulación de población» que aparece en esta revista. Es un tipo de programa que se sitúa entre una aventura y un juego de estrategia y tiene la ventaja específica de que es del tipo que aman los programadores de BASIC, ya que puede hacerse crecer y expandirse en direcciones muy diversas.

Se pueden añadir otros elementos a las «circunstancias» existentes, adornarlo con gráficos y con muchos casos más.

Piense a ver cómo puede dibujar un campo labrado. Las facilidades de edición del «Locomotive BASIC» le ayudarán mucho.

```
u10 REM ********************************
20 REM *             REINADO          *
30 REM *             -------          *
40 REM ********************************
50 REM
60 REM
70 ON BREAK GOSUB 1490
80 ON ERROR GOTO 1500
90 GOSUB 320
100 GOSUB 1190
110 IF K$="s" THEN GOSUB 480
120 INK 0,1:INK 1,24:INK 2,22:INK 3,3
130 PAPER 0:PEN 1:BORDER 1:MODE 1
140 FOR Y=1985 TO 1994+INT(RND*3)
150 CC=C:AF=0
160 GOSUB 650
170 GOSUB 740
180 GOSUB 1190
190 LOCATE 1,12:PRINT SPACE$(240);SPACE$(240):LOCATE 1,12
200 IF K$="n" THEN C=CC:GOTO 160
210 GOSUB 970
220 IF B=0 THEN 290
230 IF AF=0 THEN 280
240 GOSUB 650
250 E=4:GOSUB 1190
260 IF RND<0.75 THEN E=5:GOSUB 1190:GOTO 280
270 E=6:GOSUB 1190:GOTO 300
280 NEXT Y
```

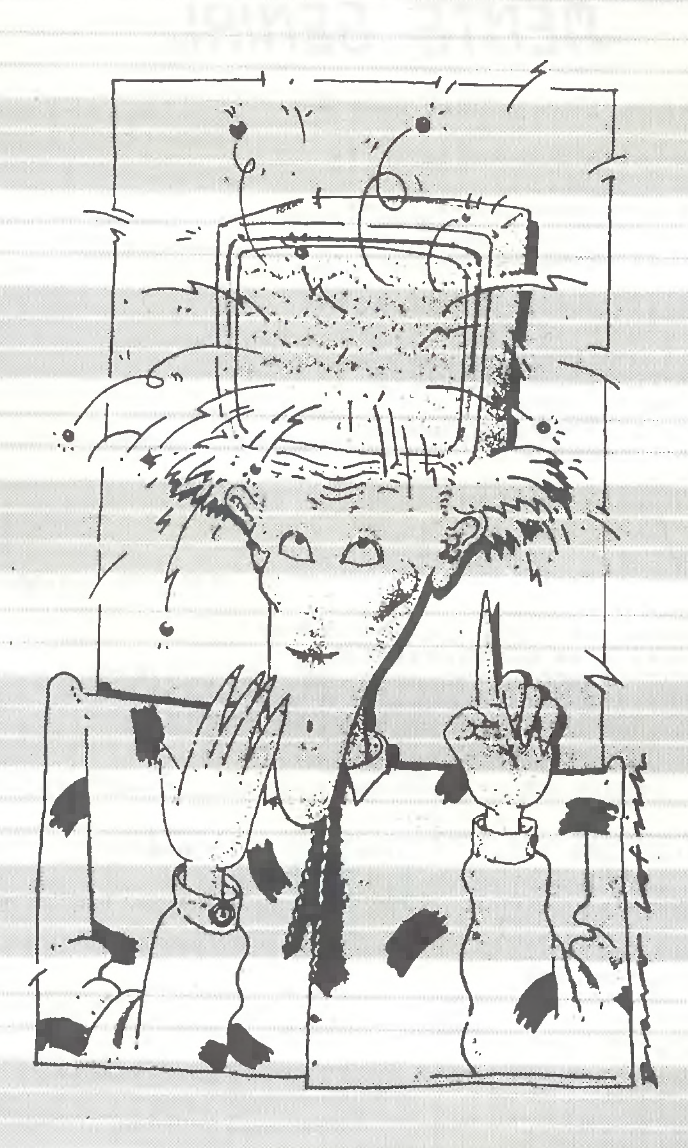
```
290 GOSUB 650
300 GOSUB 1350:'fin
310 GOTO 90
320 REM inicializacion
330 CLS
340 INK 0,1:INK 1,24:INK 2,22:INK 3,3
350 PAPER 0:PEN 1:BORDER 1:MODE 1
360 C$=CHR$(164)+" AMSTRAD USER"
370 CK$="REINADO"
380 V$="version 1.1"
390 A=INT(500+RND*500)  :'hectareas de terreno
400 B=INT(2500+RND*2500) :'poblacion
410 C=INT(75000+RND*25000):'sacos de semillas
420 E=0
430 RANDOMIZE TIME
440 LOCATE 1,1:PRINT CK$:PRINT
450 PRINT C$
460 PRINT:PRINT" Quieres instrucciones? s/n:"
470 RETURN
480 REM instrucciones
490 INK 0,13:INK 1,0
500 PAPER 0:BORDER 13:PEN 1
510 MODE 2
520 PRINT CK$;TAB(25);C$:PRINT
530 PRINT "Estás al mando de la economia de un diminuto imperio. Cada ciclo anua
l, todos los miembros de tu comunidad necesitan seis sacos de semillas para come
r de modo que puedan sobrevivir y son capaces de plantar tres sacos."
 540 PRINT "Se necesitan exactamente 10 sacos para plantar una hectarea. Si usas
menos del 60% de la tierra, parte se volvera esteril y si usas mas del 60%, incr
ementaras la tierra de labrantio."
550 PRINT
560 PRINT"Al principio de cada periodo debes decidir cuantos sacos de semillas u
saras para plantar y para dar de alimentar a la poblacion. La cantidad de semill
as que produzcan los campos dependera ";
570 PRINT " de lo que se haya plantado y de la calidad de la cosecha. Cada miemb
ro de la poblacion que no reciba los seis sacos morira de hambre y cada seis sac
os de exceso permitiran que nazca una nueva persona."
580 PRINT " Sin embargo, la poblacion no puede incrementarse infinitamente. Si m
uere mas del 30% de la poblacion, ten cuidado. Pueden intentar asesinarte!!"
590 PRINT
600 PRINT "Pulsa una tecla para seguir en respuesta a cualquier mensaje en la pa
rte inferior de la pantalla. Al final de tu mandato se te dara una puntuacion en
 funcion de tus dotes de mando. Se guarda el mejor resultado."
610 PRINT
620 PRINT TAB(15);" Pulsa una tecla para empezar - Suerte!"
630 IF INKEY$="" THEN 630
640 RETURN
650 REM muestra pantalla
660 LOCATE 13,1:PRINT CK$
670 LOCATE 13,3:PRINT "Periodo: ";y
680 PAPER 2:PEN 3
690 LOCATE 6,6:PRINT USING "&###,##,##,##";"Poblacion:          ";b
700 LOCATE 6,7:PRINT USING "&###,##,##,##";"Sacos de semillas:";c
710 LOCATE 6,8:PRINT USING "&###,##,##,##";"Terreno:            ";a
720 PAPER 0:PEN 1
730 RETURN
740 REM entrada de datos
750 LOCATE 1,10:PRINT "Introduzca el numero de sacos a:"
760 LOCATE 1,12:PRINT SPACE$(120):LOCATE 1,12
770 INPUT "Plantar:";CP:CP=INT(CP)
780 IF CP>C THEN E=1:PRINT CHR$(7):GOSUB 1190:GOTO 760
790 IF 10*A<CP THEN E=2:PRINT CHR$(7):GOSUB 1190:GOTO 760
800 IF CP<0 THEN E=3: PRINT CHR$(7):GOSUB 1190:GOTO 760
810 IF CP>3*B THEN E=7 :PRINT CHR$(7):GOSUB 1190:GOTO 760
820 C=C-CP
830 PAPER 2:PEN 3
840 LOCATE 6,7:PRINT USING "&###,##,##,##";"Sacos de semillas:";c
850 PAPER 0:PEN 1
```

PROGRAMAS

REINADO

```
860 LOCATE 1,12:PRINT SPACE$(120):LOCATE 1,12
870 INPUT "Comer:  ";CQ:CQ=INT(CQ)
880 IF CQ>C THEN E=1:PRINT CHR$(7):GOSUB 1190:GOTO 860
890 IF CQ<0 THEN E=3:PRINT CHR$(7):GOSUB 1190:GOTO 860
900 C=C-CQ
910 PAPER 2:PEN 3
920 LOCATE 6,7:PRINT USING "&###,##,##,##";"Sacos de semillas:";C
930 PAPER 0:PEN 1
940 LOCATE 1,12:PRINT SPACE$(120):LOCATE 1,12
950 PRINT "Plantar:";CP:PRINT"Comer:   ";CQ:PRINT TAB(15);"Correcto (S/N):"
960 RETURN
970 REM calculo de los nuevos valores
980 AA=CP/10:BB=CQ/6
990 IF BB>2*B THEN BB=3*B
1000 AP=AA*100/A:A=INT(A+(AP-60))
1010 IF BB/B<0.7 THEN AF=1
1020 B=INT(BB)
1030 HV=RND
1040 C=C+INT(4*CP*(HV+0.5))
1050 LOCATE 1,12: PRINT SPACE$(120):LOCATE 1,12
1060 PAPER 2:PEN 0
1070 E=INT(HV*5+1)
1080 IF E=1 THEN E$="Cosecha muy mala"
1090 IF E=2 THEN E$="Cosecha mala"
1100 IF E=3 THEN E$="Cosecha normal"
1110 IF E=4 THEN E$="Cosecha buena"
1120 IF E=5 THEN E$="Cosecha excelente"
1130 E=0
1140 LOCATE 10,15:PRINT E$
1150 FOR TT=0 TO 2500: NEXT TT
1160 PAPER 0:PEN 1
1170 LOCATE 1,15:PRINT SPACE$(40)
1180 RETURN
1190 REM mensajes
1200 IF E=0 THEN 1310
1210 IF E=1 THEN E$="No hay suficientes semillas"
1220 IF E=2 THEN E$="No hay tierra suficiente"
1230 IF E=3 THEN E$="Solo numeros positivos"
1240 IF E=7 THEN E$="Insuficiente mano de obra"
1250 IF E=4 THEN E$="TEN CUIDADO!! Intento de asesinato."
1260 IF E=5 THEN E$="Tuviste suerte, ten mas cuidado"
1270 IF E=6 THEN E$="Te han matado!!"
1280 PAPER 3
1290 LOCATE 1,24:PRINT E$
1300 E=0
1310 K$=INKEY$:IF K$="" THEN 1310 ELSE K$=LOWER$(K$)
1320 PAPER 0
1330 LOCATE 1,24:PRINT SPACE$(40)
1340 RETURN
1350 LOCATE 1,10:PRINT SPACE$(200)
1360 K$=""
1370 LOCATE 1,10
1380 S=INT(B*C*A/1E+09)
1390 IF AF=1 THEN S=0
1400 PRINT "Tu reinado ha terminado":PRINT
1410 PRINT "Se te han concedido "; s; " puntos.":PRINT
1420 IF S>H THEN H=S
1430 PRINT "La mejor puntuacion obtenida hoy es ";h:PRINT
1440 PRINT "Quieres jugar de nuevo (S/N)?:":PRINT
1450 K$=INKEY$:IF K$="" THEN 1450 ELSE K$=LOWER$(K$)
1460 IF K$="n" THEN CLS:NEW
1470 IF K$<>"s" THEN 1450
1480 RETURN
1490 STOP:'alto
1500 GOTO 70:' deteccion de errores
```

MENTE GENIAL



Este programa es una verdadera versión del tradicional juego «Mastermind». El ordenador generará un código de cuatro dígitos con un nivel de dificultad seleccionado por el usuario. Hay un máximo de diez intentos para averiguarlo y se obtiene una puntuación después de cada uno, dependiendo de la puntería. Después de introducir una prueba, pero antes de que el ordenador la mire, hay una oportunidad de cambiarla. Esto es particularmente útil debido a que permite ver la prueba en pantalla y compararla con las anteriores sin gastar un intento si se ve un error obvio.

Hay tres niveles de juego que determinan la naturaleza del código.

1. El código contiene cualquiera de los dígitos del 1 al 9, ambos inclusive pero sin repetirse.

2. Es como el nivel 1 excepto que también se usa el 0.

3. Como el anterior pero puede haber números repetidos. El más difícil.

Un punto blanco en los puntos indica que se ha acertado uno en su sitio. La posición del punto blanco no indica nada. Un círculo negro indica que se ha acertado uno pero fuera de sitio.

MENTE GENIAL

```
10 REM *** MENTE GENIAL. 1985 AMSTRAD USER ***
20 :
30 GOSUB 1000:REM ** texto introductorio **
40 :
50 GOSUB 2000:REM ** dibuja la pantalla **
60 :
70 GOSUB 3000:REM ** el juego **
80 :
90 GOTO 4000: REM ** fin/vuelve a comenzar **
100 :
1000 REM ** texto introductorio **
1010 MODE 1:INK 1,26:PAPER 0:BORDER 0:PEN 1
1020 PRINT SPC(13);"MENTE GENIAL"
1030 PRINT:PRINT SPC(13);"INSTRUCCIONES"
1040 FOR T=1 TO 5:PRINT:NEXT T
1050 PRINT"El objeto del juego es descubrir un"
1060 PRINT"codigo de cuatro cifras creado por el"
1070 PRINT"ordenador. Hay tres niveles de "
1080 PRINT"dificultad: el 1 es el mas sencillo y"
1090 PRINT"el 3 es mas dificil. En el nivel 1 se usan los digitos del 1 al 9 y n
o hay"
1100 PRINT"repetidos. En el nivel 2 puede usarse"
1110 PRINT"el 0 y en el 3 se pueden repetir."
1170 LOCATE 1,24:PRINT"SELECCIONA EL NIVEL DE JUEGO (1/2/3):"
1180 LET SK$=INKEY$:IF SK$<>"1" AND SK$<>"2" AND SK$<>"3" THEN GOTO 1180
1190 LOCATE 1,24:PRINT"PULSA CUALQUIER TECLA PARA JUGAR AL      NIVEL "+SK$
1200 CALL &BB18
1998 RETURN
1999 :
2000 CLS:BORDER 5
2010 LOCATE 23,2:PRINT CHR$(143);" ";CHR$(143);" ";CHR$(143);" ";CHR$(143);"  PU
NTOS"
2020 FOR T=348 TO 444 STEP 32:PLOT T,386:DRAW T,364:DRAW T+22,364:DRAW T+22,386:
DRAW T,386:NEXT T
2030 PLOT 332,10:DRAW 482,10:DRAW 482,394:DRAW 332,394:DRAW 332,10
2040 PLOT 333,358:DRAW 481,358
2050 WINDOW#1,3,18,2,4
2060 INK 3,0
2070 PAPER#1,13:PEN #1,3
2080 PRINT#1,"***************** MENTE GENIAL *****************";
2090 WINDOW#2,2,19,10,19
2100 PLOT 10,82:DRAW 306,82:DRAW 306,258:DRAW 10,258:DRAW 10,82
2110 PEN 1:LOCATE 2,8:PRINT"INSTRUCCIONES"
2120 LOCATE 1,22:PRINT CHR$(230)+" NUMERO CORRECTO":PRINT"  EN SITIO ERRONEO":PR
INT CHR$(231)+" NUMERO CORRECTO":PRINT"  EN SITIO CO
RRECTO"
2198 RETURN
2199 :
3000 REM *** juega ***
3010 LET TE$="":RANDOMIZE TIME
3020 FOR N=1 TO 4
3025 C(N)=INT(RND*10)
3030 NEXT N
3040 IF SK$="3" THEN GOTO 3080
3042 FOR N=1 TO 3
3044 FOR P=(N+1) TO 4
3046 IF C(N)=C(P) THEN TE$="FALLO"
3048 NEXT P
3050 NEXT N
3060 IF SK$="2" THEN GOTO 3080
3064 FOR N=1 TO 4
3068 IF C(N)=0 THEN TE$="FALLO"
3070 NEXT N
```

```
3080 IF TE$="FALLO" THEN GOTO 3000
3090 Z=1
3100 CLS#2:PRINT#2:PRINT#2,"INTENTO NO.";Z:PRINT#2:PRINT#2,"?";
3110 FOR N=1 TO 4
3120 LET IN$=INKEY$:IF SK$="1" AND IN$<"1" OR SK$="1" AND IN$>"9" THEN 3120 ELSE
 IF IN$<"0" OR IN$>"9" THEN 3120
3130 U(N)=VAL(IN$):PRINT#2,U(N);
3140 NEXT N
3150 FOR N=1 TO 4:PRINT#2:NEXT N
3155 PRINT#2,"ES ESTE EL QUE    QUIERES? (S/N)  "
3160 IN$=INKEY$:IF UPPER$(IN$)="N" THEN CLS#2:GOTO 3100
3170 IF UPPER$(IN$)<>"S" THEN GOTO 3160
3180 LET U$="":FOR N=1 TO 4:LET U$=U$+RIGHT$(STR$(U(N)),2):NEXT N
3190 LOCATE 22,2+(Z*2):PRINT U$
3200 REM se compara la entrada y la salida
3210 LET COMP$=""
3220 FOR N=1 TO 4:P(N)=0:Q(N)=0:NEXT N
3230 FOR N=1 TO 4
3235 IF U(N)=C(N) THEN COMP$=COMP$+CHR$(231):P(N)=N:Q(N)=N
3240 NEXT N
3250 IF LEN(COMP$)=4 THEN LOCATE 33,(2+(2*Z)):PRINT COMP$:GOTO 3350
3260 FOR T=1 TO 4
3270 IF P(T)<>0 THEN GOTO 3310
3280 FOR N=1 TO 4
3285 IF Q(N)<>0 THEN GOTO 3300
3290 IF T<>N AND U(N)=C(T) THEN COMP$=COMP$+CHR$(230):Q(N)=N:N=4
3300 NEXT N
3310 NEXT T
3320 LOCATE 33,(2+(2*Z)):PRINT COMP$
3330 IF Z<10 THEN LET Z=Z+1: GOTO 3100
3340 TE$="":IF LEN(COMP$)<4 THEN TE$="FALLO"
3350 FOR N=1 TO 4:IF MID$(COMP$,N,1)=CHR$(230) THEN TE$="FALLO":NEXT N
3360 CLS#2:PRINT#2:IF TE$="FALLO" THEN PRINT #2," MALA SUERTE!" ELSE PRINT #2,"
BIEN HECHO!"
3370 PRINT#0:LOCATE 22,2:C$="":FOR N=1 TO 4:C$=C$+RIGHT$(STR$(C(N)),2):NEXT N:PR
INT C$
3380 FOR T=348 TO 444 STEP 32:PLOT T,386:DRAW T,364:DRAW T+22,364:DRAW T+22,386:
DRAW T,386:NEXT T
3998 RETURN
3999 :
4000 REM vuelve a ejecutar o termina
4010 PRINT #2:PRINT#2:PRINT#2,"OTRO JUEGO? (S/N)  "
4020 IN$=INKEY$:IF LEN(IN$)=0 THEN 4020 ELSE IF UPPER$(IN$)="N" THEN 4999
4030 IF UPPER$(IN$)<>"S" THEN 4020
4040 CLS#2
4050 PRINT#2,"JUGASTE AL NIVEL ":PRINT#2,SK$+".":PRINT#2,"QUIERES CAMBIAR?  (S/N
)"
4060 IN$=INKEY$:IF LEN(IN$)=0 THEN 4060 ELSE IF UPPER$(IN$)="N" THEN 4998
4070 IF UPPER$(IN$)<>"S" THEN 4060
4080 PRINT#2:PRINT#2,"INTRODUCE EL NIVEL"
4090 IN$=INKEY$:IF LEN(IN$)=0 THEN 4090 ELSE IF IN$<"1" OR IN$>"3" THEN 4090
4100 SK$=IN$
4110 PRINT#2:PRINT#2,"NIVEL "+SK$:PRINT#2,"SELECCIONADO":FOR T=1 TO 1000:NEXT T
4998 GOTO 50
4999 CALL &0
5000 REM fin de mente genial
```





AMSTRAD CPC128

En fecha aún por determinar, coincidiendo con su lanzamiento en el mercado británico e internacional, comenzará a distribuirse en España el AMSTRAD 6128. El nuevo modelo marca una línea de total continuidad con el popular CPC 464 y presenta el indudable atractivo de ofrecer una memoria RAM de 128 K. En el próximo número de nuestra Revista estaremos en condiciones de presentar una exhaustiva información sobre el AMSTRAD 6128, características y modo de operar. Un paso adelante.



Lápiz óptico

Los poseedores de los CPC con monitor en color pueden incorporar en sus programas un mecanismo de selección de opciones muy cómodo: basta apuntar con el lápiz a la pantalla para que la máquina elija la opción adecuada. Eso si disponen del nuevo lápiz óptico LP-1.

El lápiz, realmente compacto, se incorpora en la toma de **joystick**, e incluye una cinta con unos programas de demostración. En las instrucciones del programa se indica cómo se puede usar en otros programas.

Realizado por **Trojan**, que ya sorprendió a los usuarios de Spectrum con un lápiz sencillo y eficaz, este producto comparte estas características; su único inconveniente es que no es compatible con el **port** serie y el sintetizador de voz. Una limitación más seria es que no funciona correctamente con el monitor monócrono.



Sintetizador de voz, con amplificador

Aunque la serie CPC se caracteriza por una gran integración de periféricos en la unidad principal, existen dispositivos especializados que sólo justifican su inclusión como accesorios. Una de estas categorías es la de los sintetizadores de voz, de los que **Amstrad** acaba de presentar su modelo SSA-1.

Con él se puede hacer hablar al ordenador, a través de su amplificador estéreo. Para que no falte nada, la unidad incorpora dos pequeños altavoces, aunque la calidad mejorará mucho si utilizamos una cadena hi-fi.

La unidad incluye el **software** para controlar el sintetizador en un **cassette**. Una vez cargado, el ordenador reconoce varios comandos extras, mediante los que se le puede indicar que repita lo que se imprima en pantalla, que pronuncie una frase,... Se puede programar su pronunciación mediante alófonos, con lo que la pronunciación del castellano (con un leve acento británico) es perfectamente programable.





Al vuelo

Fue una tormenta de verano.

Empezaron a oírse voces indignadas: «¿Que un **joystick** va a tener un arancel mínimo de 15.000 ptas.?» Después vinieron las rectificaciones: la medida afecta sólo a los ordenadores de menos de 64 K.

Una petición a la Junta de Aranceles por parte de un fabricante español de ordenadores domésticos, junto con un error administrativo, desencadenó la tormenta. El arancel penaliza notablemente a los equipos más baratos, frente a los de tamaño medio. Así, muchas compañías piensan seriamente en una «planta de fabricación», donde se metan las máquinas (llegadas de Inglaterra) en bonitas cajas; otros importadores dejarán caer en el interior del ordenador más chips de memoria, otra manera de saltarse el arancel. En cualquier caso, la aplicación del régimen arancelario durará sólo hasta marzo, ya que en este mes se adopta la legislación comunitaria en este sector.

La medida no afectará, naturalmente, al nuevo modelo CPC 6128, que es un ordenador de 128K de RAM, pero sí incidirá en el CPC 464.

No obstante, AMSTRAD ESPAÑA no hará repercutir este aumento de costes en el precio final de venta al usuario, que se mantendrá invariable.

UN ORDENADOR MUY MUSICAL (1)

El Amstrad representa muchas cosas distintas para diferente gente y entre sus varias capacidades, tiene un avanzado generador de sonidos que puede producir cualquiera, desde un «disparo láser» a música de teclado. En este primer artículo de una serie destinada a la música vamos a ver uno de los comandos responsables de los sonidos desde BASIC.

NO se asuste al ver la palabra «música», no hace falta saber leer una partitura para poder hacer sonar al CPC 464 (o al 664, aquellos afortunados que lo tengan), además se irán dando los conceptos básicos según haga falta. Pero antes de empezar veamos algunos datos básicos.

Dentro del **Amstrad** se encuentra un circuito integrado encargado de generar el sonido que se llama AY-3-8912 fabricado por **General Instruments.** Es el responsable de crear todo el sonido. ¿Pero qué es el sonido? Esta pregunta puede parecer una tontería, pero no es así ya que cualquier sonido posee propiedades muy complejas y una comprensión de qué es, ayudará a usar el circuito al máximo.

Los tres componentes básicos del sonido son: duración, amplitud y tono. Los tres forman las raíces de la generación del sonido en los ordenadores, de modo que los veremos uno a uno.



Duración

Se refiere a la duración de un sonido en el tiempo. Es importante ya que determina el ritmo de una composición y la fuerza con que suena la nota. Este último punto puede parecer raro, pero un sonido o un ruido pueden oírse más fuerte o más débil, dependiendo de la duración. Una nota muy corta sonará más suave que una más larga.

En la notación musical la duración juega un papel muy importante y luego veremos cómo se le indica a un músico. También constituye uno de los componentes fundamentales de la primera instrucción BASIC que vamos a ver, *SOUND.* Pero antes veamos los otros dos factores que intervienen en la creación de sonido.

Amplitud

Ya hemos comentado el tema de la amplitud al hablar de la duración. Simplemente es el volumen con que el oído percibe el sonido. En el CPC 464 y en el 664 hay dos sistemas de alterar el volumen. Primero, por medio del mando situado a la derecha de la máquina y también bajo control de un programa. Esto parece ser el final de esta historia. La amplitud es simplemente el volumen de un ruido, ¿no?, ¡pues no es sólo eso!



Música

Todos sabemos lo que es la música, pero si nunca ha tocado un instrumento o ha leído una partitura, todos esos extraños símbolos le harán creer que aprender árabe es fácil. Pero no

se rinda. Aprender los fundamentos de la música no es difícil y sería una tontería el no obtener el máximo de su ordenador. Asumiremos aquí que se sabe la notación musical básica. Si no es así, léase las primeras páginas de cualquier libro de teoría musical.

Para empezar el rango de notas posibles es el siguiente:

DO DO#/REb RE RE#/MIb MI/FA FA#/SOLb/SOL/SOL#/LAb/LA/LA#/SIb SI.

En este tipo de música existen doce notas, estando separadas entre sí por un semitono. En general, el espacio entre notas (*DO, RE,* etc.) es un tono. La indicación de cuando una nota sube o baja un semitono se muestra normalmente por medio de un sostenido (# o un bemol (*b*) respectivamente.

Como todo en la vida, las cosas no son tan sencillas. Dentro de un sonido dado puede haber otras amplitudes que no son tan aparentes para la persona que está escuchando. Si piensa acerca del sonido de un instrumento en particular, verá que cada uno se diferencia de los demás. Esto es debido a que el volumen varía dentro de la nota que se toca. Volveremos a tratar este tema cuando examinemos las instrucciones que alteran la envolvente *(ENV* y *ENT).* El último componente básico es el tono.



Tono

El tono de una nota indica simplemente si el sonido es grave o agudo. O diciéndolo de otra forma, el tono es la frecuencia del sonido. Como ya sabe, el sonido viaja en forma de ondas y el

T A L

número de veces que una onda en especial vibra por segundo determina su frecuencia. Normalmente se mide en Hertzios (Hz), que es el número de ciclos por segundo. El oído humano no tiene una capacidad infinita, sino que sólo oye dentro de un rango determinado de frecuencias, aproximadamente entre 20 y 20.000 Hz.

Un ejemplo claro de esta limitación son los silbatos de perros. Debido a que los perros pueden oír a frecuencias superiores que los humanos, son capaces de detectar el ruido del silbato, aunque no lo podamos escuchar nosotros. De modo que si no podemos oír un sonido, no quiere decir que no exista.

Para los fines de esta serie de artículos, el tono se referirá a las frecuencias de la escala musical usada en occidente. En el teclado de un piano se tiene un sonido fijo que puede ser tocado a distintos tonos, altos o bajos, dependiendo de la tecla que se pulse. Las notas ascienden desde la izquierda del teclado (una nota grave) hasta la derecha (una aguda). Estas frecuencias del sonido se fijan a intervalos regulares.

El CPC 464 permite al usuario controlar el tono de un sonido de un modo muy preciso por medio de la sentencia *ENT*. Pero antes nos tomaremos un respiro para ver qué es posible hacer con el ordenador y ver la notación musical.

Las excepciones (siempre existen) son intervalos entre *SI* y *DO* y entre *Mi* y *FA*, que son semitonos. La lista dada anteriormente corresponde a los doce semitonos. No, no hay un error, aunque hay más de doce notas distintas, varias corresponden al mismo semitono. Por ejemplo, *RE*# y *MIb* son el mismo exactamente.

Una escala se compone de ocho notas en el siguiente orden: tono, tono, tono, semitono, tono, tono, tono, semitono. Todas las notaciones musicales se dan en esta escala. El ejemplo clásico son las teclas blancas del piano, que cuando se tocan en esa secuencia (desde la *do*) dan la secuencia del *do* mayor.

La diferencia entre el sonido del *do* inferior y del *do* situado en la parte superior de la escala se llama octava (8 notas), y por tanto todas las escalas empiezan y acaban en la misma nota. La nota que termina la escala es, además, la que empieza la siguiente. No una hay razón particular para usar esta notación, pero es la más usada en España y se ha hecho durante cientos de años.

Con esto ya es suficiente en cuanto a lecciones de música. Ahora sus dedos deben estar deseando tocar el teclado, por lo que examinaremos la primera sentencia BASIC: **SOUND.**



La instrucción SOUND

Lo más terrorífico de usar esta instrucción es el número de parámetros diferentes que se pueden usar. Esto hace que muchas veces la gente se asuste y no lo use, ya que parece muy complicado. La realidad es todo lo contrario. La sentencia **SOUND** es muy fácil de usar una vez que ves las funciones de cada parte de la sentencia.

Esta instrucción tiene capacidad para manejar todos los elementos del sonido que hemos visto hasta ahora de un modo simple o, lo que es más importante, desde BASIC.

El comando al completo es como sigue:

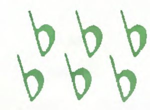
SOUND C, T, D, V, VE, TE, PR



Como se puede ver hay siete parámetros distintos, pero sólo son imprescindibles los dos primeros. Los cinco restantes son opcionales (D a PR). El primer parámetro, «C», es el canal que se va a usar y su estado.

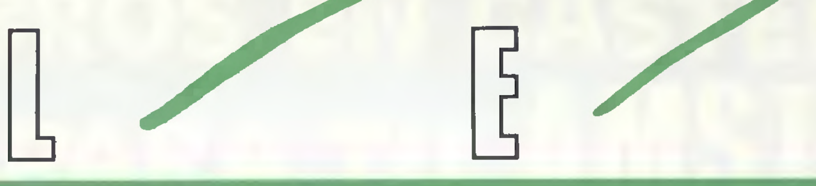
Este parámetro tiene una variedad de efectos y por el momento sólo vamos a ver qué son estos canales y cómo sacar sonido a través de ellos.

Si piensa en la música que normalmente se oye, verá que muchas veces se toca más de una nota a la vez (por ejemplo). Esto permite la creación de diferentes acordes tan usuales en todo tipo de música. El CPC 464 (y el 664 evidentemente) tienen esta posibilidad haciendo que por cada canal suene una nota distinta. El máximo disponible son tres notas, ya que la máquina tiene tres canales.



El segundo parámetro, «P», también es necesario, ya que le dice al circuito encargado de generar el sonido qué tono tiene que sonar. No confunda este valor con los números de frecuencias dados en la guía del usuario. Refiérase siempre al número del tono (período) de la nota que quiere producir. Use el ejemplo anterior, pero cambie el número del tono.

Una cosa curiosa es que la nota se hace más grave cuanto más alto es el número del tono. Esto es correcto y es una característica que se olvida fácilmente.





El tercer parámetro es la duración, «D». Este es el primero de la lista de valores opcionales que se le pueden dar. Si no se especifica nada, el sonido producido tendrá una duración de un veinteavo de segundo, que es el valor por defecto.

Los valores admisibles para este parámetro son todos aquellos comprendidos entre −32.768 y +32.767. Este parámetro puede ser usado de tres modos distintos. En el primero se indica el tiempo que suena el tono. En este caso debe ser un entero positivo y cada unidad es una centésima parte de un segundo. De modo que si D es 100, el tono sonará durante un segundo. Los otros dos métodos de usar este parámetro los veremos posteriormente, ya que están relacionados con el parámetro ENV.



Por último, este mes veremos el cuarto parámetro, «V», que es el volumen o amplitud. En la guía del usuario se indica que los valores posibles son del 0 al 15, pero si se introduce uno del 8 al 15, lo que se hace es repetir los del 0 al 7.

La amplitud se incrementa desde un silencio total (0) hasta el volumen máximo (7). Esto se ve en el siguiente programa, en el que se va aumentando el volumen por medio de un bucle **FOR-NEXT.**

```
10: FOR volumen=0 to 7.
20: SOUND 2, 3189, 30, volumen.
30: NEXT.
```

Y con esto concluye esta introducción al sonido y a la música. Juegue con los parámetros que hemos comentado hasta que esté satisfecho y haya comprendido sus efectos individuales. El mes que viene empezaremos a tocar música con el Amstrad y veremos en detalle el resto de la instrucción **SOUND** y las instrucciones que controlan la envolvente, **ENV** y **ENT.**

Un ejemplo complejo de programación de música

```
10 REM el formato de los datos es XXX, un numero de tres bits que indica el cana
l
20 REM                              nota, dos caracteres que indican la nota
30 REM                              oct, un numero que indica la octava
40 REM                              len, un numero que indica la longitud
50 REM 1 la menor, 9 la maxima
60 unit=4.5
70 ENV 1,UNIT,2,1,UNIT,-2,1
80 ENV 2,UNIT,2,1,1,0,2*UNIT,UNIT,-2,1
90 ENV 3,UNIT,2,1,1,0,4*UNIT,UNIT,-2,1
100 ENV 4,UNIT,2,1,1,0,6*UNIT,UNIT,-2,1
110 ENV 5,UNIT,2,1,1,0,10*UNIT,UNIT,-2,1
120 ENV 6,UNIT,2,1,1,0,14*UNIT,UNIT,-2,1
130 ENV 7,UNIT,2,1,1,0,18*UNIT,UNIT,-2,1
140 ENV 8,UNIT,2,1,1,0,22*UNIT,UNIT,-2,1
150 ENV 9,UNIT,2,1,1,0,30*UNIT,UNIT,-2,1
160 ENV 10,1,0,8*UNIT
170 DIM PERIODO(12)
180 PERIODO(1)=3822:PERIODO(2)=3608:PERIODO(3)=3405:PERIODO(4)=3214
190 PERIODO(5)=3034:PERIODO(6)=2863:PERIODO(7)=2703:PERIODO(8)=2551
200 PERIODO(9)=2408:PERIODO(10)=2273:PERIODO(11)=2145:PERIODO(12)=2025
210 NOTES$="  c c+d d+e f f+g g+a a+b "
230 READ A$
240 WHILE A$<>"*"
250 SPOINT=1
```

Un ejemplo complejo de programación de música

```
260 WHILE SPOINT<LEN(A$)
270 CHANA=0:CHANB=0:CHANC=0
280 CHAN$=MID$(A$,SPOINT,3):SPOINT=SPOINT+3
290 IF VAL(LEFT$(CHAN$,1)) THEN CHANA=1
300 IF VAL(MID$(CHAN$,2,1)) THEN CHANB=2-8*(CHANA<>0):CHANA=CHANA-16*(CHANA<>0)
310 IF VAL(RIGHT$(CHAN$,1)) THEN CHANC=4-8*(CHANA<>0)-16*(CHANB<>0):CHANA=CHANA-
32*(CHANA<>0):CHANB=CHANB-32*(CHANB<>0)
320 IF VAL(LEFT$(CHAN$,1)) THEN NOTE$=MID$(A$,SPOINT,4):SPOINT=SPOINT+4:GOSUB 39
0:SOUND CHANA,PERIODO,0,0,ENVEL
330 IF VAL(MID$(CHAN$,2,1)) THEN NOTE$=MID$(A$,SPOINT,4):SPOINT=SPOINT+4:GOSUB 3
90:SOUND CHANB,PERIODO,0,0,ENVEL
340 IF VAL(RIGHT$(CHAN$,1)) THEN NOTE$=MID$(A$,SPOINT,4):SPOINT=SPOINT+4:GOSUB 3
90:SOUND CHANC,PERIODO,0,0,ENVEL
350 WEND
360 READ A$
370 WEND
380 END
390 REM calcula los parametros de la nota
400 NOTE=INSTR(NOTES$,LEFT$(NOTE$,2))
410 OCTAVE=VAL(MID$(NOTE$,3,1))
420 ENVEL=VAL(RIGHT$(NOTE$,1))
430 BASE=PERIODO(NOTE\2)
440 PERIODO=BASE/(2* OCTAVE)
450 RETURN
460 DATA "101g 44g 24111d 54f 34b 24111d 54f 34b 24111e 54f 34b 24"
470 DATA "101d 54c 34111c 54g 34e 34111c 54g 34e 34111e 54g 34e 34"
480 DATA "101f 54c 34111a 54a 34f 34111c 64a 34f 34111a 54a 34f 34"
490 DATA "101a 54c 34111g 54g 34e 34111g 56g 34e 34011g 34e 34"
500 DATA "101a 54g 24111b 44f 34b 24111b 44f 34b 24111a 54f 34b 24"
510 DATA "101g 54c 34111c 54g 34e 34111c 54g 34e 34111e 54g 34e 34"
520 DATA "101e 54d 34111d 54c 44f+34111e 54c 44f+34111d 54c 44f+34"
530 DATA "101e 54g 24111d 54f+44b 24111b 44f+34b 24111g 44f+34b 24"
540 DATA "101g 44g 24111d 54f 34b 24111d 54f 34b 24111e 54f 34b 24"
550 DATA "101d 54c 34111c 54g 34e 34111c 54g 34e 34111e 54g 34e 34"
560 DATA "101f 54c 34111a 54a 34f 34111c 64a 34f 34111a 54a 34f 34"
570 DATA "101a 54c 34111g 54g 34e 34111g 56g 34e 34011g 34e 34"
580 DATA "101a 54g 24111b 44f 34b 24111b 44f 34b 24111a 54f 34b 24"
590 DATA "101g 54c 34111c 54g 34e 34111c 54g 34e 34111e 54g 34e 34"
600 DATA "101e 54g 24111d 54f 34b 24111e 54f 34b 24111d 54f 34b 24"
610 DATA "101e 54c 34111c 54g 34e 34111g 36e 36c 56"
620 DATA "101e 56c 34011a 34f 34111c 56a 34f 34011a 34f 34"
630 DATA "101a 46c 34011g 34e 34111g 46g 34e 34011g 34e 34"
640 DATA "101g 54g 24111d 54f 34b 24111e 54f 34b 24111f 54f 34b 24"
650 DATA "101e 54g 24111d 54g 34e 34111c 56g 34e 34011g 34e 34"
660 DATA "101e 56c 34011g 34e 34111c 56g 34e 34011g 34e 34"
670 DATA "101a 46c 34011g 34e 34111g 46g 34e 34011g 34e 34"
680 DATA "101f+44g 24111g 44f 34b 24111a 44f 34b 24111b 44f 34b 24"
690 DATA "101d 54c 34111c 54g 34e 34111c 56g 34e 34011g 34e 34"
700 DATA "101e 56c 34011g 34e 34111c 56g 34e 34011g 34e 34"
710 DATA "101a 46c 34011g 34e 34111g 46g 34e 34011g 34e 34"
720 DATA "101g 54g 24111d 54f 34b 24111e 54f 34b 24111f 54f 34b 24"
730 DATA "101e 54g 24111d 54g 34e 34111c 56g 34e 34011g 34e 34"
740 DATA "101e 56c 34011g 34e 34111c 56g 34e 34011g 34e 34"
750 DATA "101a 46c 34011g 34e 34111g 46g 34e 34011g 34e 34"
760 DATA "101f+44g 24111g 44f 34b 24111a 44f 34b 24111b 44f 34b 24"
761 DATA "101d 54c 34111c 54g 34e 34111c 56g 36e 36"
770 DATA "*"
```

LIBROS EN CASTELLANO PARA TU AMSTRAD



Manual de Referencia Basic para el Programador
La más autorizada y completa guía para programar en Locomotive Basic.
3.400.— Pts.

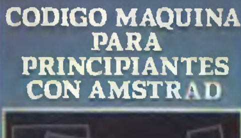


Juegos Sensacionales para AMSTRAD
Listados completos de 27 estupendos juegos de muy diversos estilos.
1.950.— Pts.



Programando con AMSTRAD
Fundamental para el usuario principiante.
Ameno y repleto de ejemplos.
2.400.— Pts.

40 Juegos Educativos
Listados completos (matemáticas, geografía, música, etc.) para aprender divirtiéndose.
1.950.— Pts.





Código máquina para principiantes con AMSTRAD
Ideal para iniciarse en el código máquina del Z80 y en el sistema operativo del AMSTRAD.
2.100.— Pts.



Hacia la Inteligencia Artificial con AMSTRAD
Convierta su AMSTRAD en un compañero inteligente
1.500.— Pts.



Música y Sonidos con AMSTRAD
Programe música y efectos sonoros y convierta su AMSTRAD en un sintetizador.
1.200.— Pts.

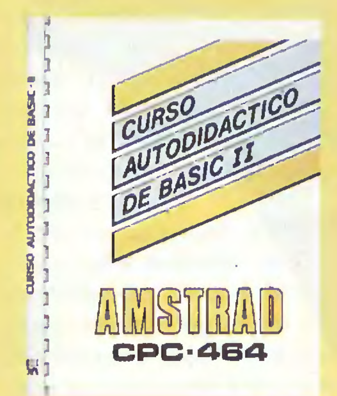


Programación de Basic con AMSTRAD.
Imprescindible para el principiante y eficaz herramienta para el programador avanzado.
2.100.— Pts.



Técnicas de Programación de Gráficos en el AMSTRAD
Este libro enseña a aprovechar las excelentes funciones gráficas del AMSTRAD, con múltiples ejemplos.
1.950.— Pts.





Curso Autodidáctico de Basic I y II
Un completo y estructurado Curso de Basic apoyado con numerosos ejemplos y acompañado de cassettes.
2.900.— Pts. cada volumen

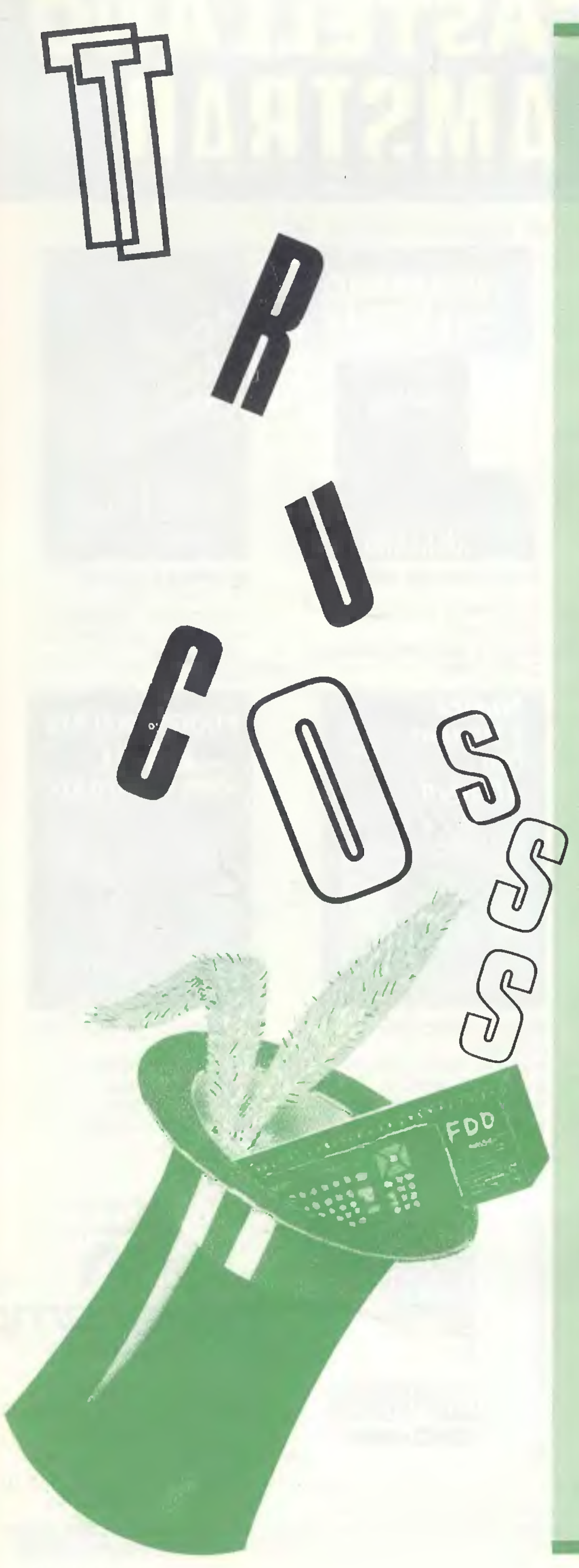
indescomp (tm)
PUBLICACIONES

Avd. del Mediterráneo, 9
Telfs.: 433 45 48 — 433 48 76
28007 MADRID

Delegación en Cataluña:
C/. Tarragona, 110 — Telf. 325 10 58
08015 BARCELONA

DE VENTA EN EL CORTE INGLES
Y TIENDAS ESPECIALIZADAS

(tm) Marca Registrada por el Grupo Indescomp.



Convierta su 664 en un 464

¿Un paso retrógado? Veamos aquí por qué no siempre es así

Los afortunados que han comprado un CPC 664 habrán comprado una buena cantidad de cintas a la espera de que aparezcan los primeros programas en disco. Desafortunadamente pueden haber descubierto que muchos de estos juegos (Codename MAT y Pijamarama, por ejemplo) no cargan, sino que dan un error «MEMORY FULL».

La razón de esto no es debida a un problema de incompatibilidad entre el 664 y el 464 (lo mismo sucede en un 464 con discos). Sino que se debe a que el sistema operativo del disco usa 1284 bytes para su uso propio. En un 464 con discos la solución es simple. No encienda el disco antes que el ordenador. Con un 664, la ROM del disco siempre se conecta al encender la máquina. No hay ninguna solución sencilla por hardware para que el BASIC recupere la memoria usada por el disco.

Pero no se preocupe, la ayuda está aquí si teclea el pequeño programa listado a continuación y lo salva en disco. Una vez esto, cuando quiera cargar un programa de cinta, coja el disco y ejecute el programa. Se encontrará con el mensaje «BASIC 1.1» (o BASIC 1.0 si utiliza un 464 con discos). Si ahora hace un PRINT HIMEM obtendrá un 43903 en lugar de un 42619 que es lo que da con el disco funcionando. Un segundo efecto de esta orden es que ninguna de las órdenes del disco (|TAPE, |DIR, etc.) funcionará. La máquina funcionará como si fuese un 464.

```
190 MEMORY &7FFF
200 ENTRY=&8000
210 SUM=0
220 FOR ADDR=0 TO &3E
230 READ BYTE$
240 POKE ENTRY+ADDR,VAL("&"+BYTE$)
250 SUM=SUM+VAL("&"+BYTE$)
260 NEXT ADDR
270 IF SUM <> 6058 THEN PRINT "error en los datos":END
280 CALL ENTRY
290 NEW
300 PRINT HEX$(SUM)
310 DATA CD,C8,BC,7A,B3,20,05,06,00,11,06,C0,ED,53,3C,80
320 DATA ED,43,3E,80,21,1C,80,0E,FC,CD,16,BD,2A,3C,80
330 DATA ED,4B,3E,80,48,3E,C9,32,CB,BC,22,39,80,79,32,3B,80
340 DATA 11,40,00,21,FF,AB,DF,39,80,00,00,00,00,00,00,00
```

Uso avanzado del INKEY$

Normalmente se suele utilizar la instrucción INKEY$ para leer caracteres del teclado, pero su uso suele ser bastante sencillo y no todo lo sofisticado que sería de desear. En la pequeña rutina que se lista abajo se considera que en PT$ hay almacenada una lista de todos los caracteres que se pueden aceptar como datos de entrada. Esta lista puede ser un solo carácter (un espacio, por ejemplo) o un conjunto de caracteres «YN», «ABCDE», «12345», etc.

La rutina espera a que se pulse una tecla y luego mira si el carácter está incluido en la lista. Si no está incluido lo ignora y si está, devuelve la posición dentro de PT$ al volver, por medio de la variable PL. Los mayores beneficios de esto es que no se aceptan caracteres no válidos y el carácter aceptado es convertido en un número que se puede usar directamente por un ON PL GOTO (o GOSUB). Esto reducirá el tamaño del programa y hará más sencilla la programación.

```
3000 PL=0:WHILE PL=0:A$=INKEY$:IF A$>"" THEN PL=INSTR(PT$,UPPER$(A$))
3010 WEND:RETURN
```

Los saltos de líneas en las impresoras

Si usted ha comprado la impresora Indescomp junto el cable de conexión, no habrá tenido ningún problema para ponerla en marcha ya que viene especialmente preparada para ello. Pero si ha comprado otra impresora que no venga preparada especialmente para el Amstrad, puede haber tenido algún problema debido a que imprima a doble espacio (dejando una línea en blanco entre dos líneas impresas). Este es un problema muy común con determinado tipo de impresoras. En primer lugar, compruebe los interruptores y especialmente uno que debe ir etiquetado «AUTO LINE FEED» (en el manual de la impresora se le indicará dónde está situado) que debe estar en la posición OFF, es decir, que no haga salto de línea automático. Esto se debe a que el ordenador envía dos caracteres para indicar el salto de línea: el CR (retorno del carro) y el LF (salto de línea). Si dicho interruptor está colocado erróneamente en la impresora, ésta hará a su vez otro salto y dejará una línea en blanco. Si pese a colocar los interruptores en la posición adecuada sigue dejando una línea en blanco, corte el cable que va a la patilla 14 del conector de la impresora (estos conectores suelen ir numerados, pero hágalo con cuidado) ya que hay muchas impresoras que necesitan que esta patilla esté en circuito abierto (sin conectar a nada) para que hagan salto de línea por su cuenta.





Libro: Programando con Amstrad
Autor: Ian Sinclair
Editorial: Amstrad
Páginas: 182

Este libro está orientado a los usuarios noveles del ordenador que necesitan empezar a manejarlo sin ninguna experiencia previa y que desean adentrarse en las profundidades de la programación en *BASIC*. Para ello empieza con un capítulo en el que explica como instalar el ordenador, la conexión a la red (curiosamente se hace referencia a los enchufes ingleses, ligeramente distintos a los nuestros, aunque las explicaciones son válidas), al monitor o al televisor y el encendido. A continuación se pasa revista al teclado, explicando detalladamente el uso de las teclas que no son normales en otros aparatos como el la máquina de escribir (*ENTER, CTRL*, etc). Una vez visto ésto se empiezan a dar las primeras nociones de programación en *BASIC*, pasando por la instrucción y sus múltiples usos hasta llegar a manejos complejos como es la realización de listados encolumnados.

En siguientes capítulos se van examinando más instrucciones del *BASIC* y su uso en la realización de programas. Se ven los bucles por medio del *FOR... NEST*, las subrutinas y funciones, la realización de menús (denominados «menúes» en el libro) y otras instrucciones de uso general.

En capítulos siguientes se empiezan a tratar temas más concretos y complejos como son los archivos en disco y cinta. En este capítulo además de explicar el funcionamiento desde *BASIC*, se hace una descripción del sistema de grabación utilizado, lo que resulta realmente interesante, ya que permite comprender el cómo y porqué del funcionamiento y por tanto su uso más óptimo.

Los tres últimos capítulos del libro están dedicados a lo que quizás es más llamativo. Los gráficos y sonido. A lo largo de las páginas se nos enseñan todas las instrucciones que nos permiten realizar dibujos y sonido, pasando de las más sencillas a manejos realmente complejos, de modo que se pueden acabar realizando programas espectaculares en este aspecto.

Todos los temas que hemos comentado están acompañados de gran cantidad de programas de ejemplo en los que se detalla el uso de todas las instrucciones. Estos programas son pequeños y están destinados a enseñar el funcionamiento de las instrucciones y no son útiles en sí mismos. También hay que señalar la existencia de cuatro apéndices de gran utilidad para todos los usuarios.

El libro es muy útil como manual introductorio al manejo de la máquina y resultará de gran utilidad para todos aquellos que se enfrenten por primera vez con un ordenador o para los que quieran perfeccionar sus conocimientos.



Libro: Juegos sensacionales para Amstrad
Autor: J. Gregory
Editorial: Amstrad
Páginas: 210

Al contrario de otros libros de carácter basicamente educativo, éste se dedica a hacer un compendio de diversos juegos (y algunas utilidades) de todo tipo, de modo que el usuario pueda disponer de una amplia gama de juegos a un precio módico.

El libro incluye 27 programas, de los que 24 son juegos y los tres restantes son utilidades generales. Cada programa, aparte del listado en sí, incluye un breve comentario en el que se explica brevemente su funcionamiento y como usarlo una vez que se ha tecleado. Para facilitar el manejo del libro, se ha dividido en varias seccio-

nes, cada una de las cuales engloba un tipo de juegos. Los primeros son juegos de texto, en los que el ordenador «conversa» con el usuario y todo el juego se desarrolla en forma de texto que se muestra por la pantalla y se introduce por el teclado. La segunda sección se refiere a los juegos que se consideran clásicos en el mundo de la informática, debido a que fueron de los primeros en aparecer y se difundieron por todo el mundo. En el tercer grupo están los juegos para dos jugadores. En éstos el ordenador se limita a hacer de árbitro y poner el tablero, como es el caso de un ajedrez o de unas damas.

Después se encuentran tres grupos de programas que se basan en juegos de la vida real, que son recreados por el ordenador. Tal es el caso de los juegos de dados, de cartas y mentales (donde nos podemos encontrar un mastermind). El grupo siguiente es el de los juegos de acción.

El último grupo —dedicado propiamente a los juegos— es el de los educativos, en el que podemos encontrar juegos de palabras y aritmética. Al final viene también un grupo especial dedicado a las utilidades. Estas son tres, de uso general, que harán las delicias de más de un usuario. La primera es un editor de caracteres que permite redefinir el juego de caracteres que nos aparece en pantalla de modo que podamos programar en gótico, o hacer gráficos a base de letras. La segunda utilidad es un editor de pantallas que nos permite hacer dibujos en pantalla como si de una hoja de papel se tratase, para almacenarlas y recuperarlas cuando queramos. La última utilidad es un monitor que nos permite examinar la memoria del ordenador, viendo sus contenidos y modificando los datos que queramos.



Libro: Amstrad CPC-464 programación avanzada
Autor: Mark Harrison
Editorial: ra-ma
Páginas: 147

Para empezar hay que decir que éste no es un libro para principiantes, como su propio nombre indica, ni para aprender a programar en lenguaje máquina; ya que todos los capítulos se refieren al *BASIC*. Básicamente lo que pretende el autor es enseñar tecnicas avanzadas de programación a gente que ya tiene un conocimiento previo de las instrucciones del *BASIC* y de su manejo.

El primer capítulo contiene una descripción del *Amstrad*, tanto a nivel externo como a nivel de componentes internos. También en este capítulo se da un breve resumen (una o dos líneas como máximo) de todas las instrucciones que admite, junto con un esquema básico de su uso. Esta descripción continúa en el capítulo dos, dedicado al funcionamiento de las cadenas alfanuméricas, que también da, bastante resumido, todo el conjunto de operaciones que se pueden hacer con ellas.

Ya en el capítulo tres se empiezan a explicar técnicas avanzadas de entrada y salida, como coger datos del teclado, manejos especiales del *PRINT*, métodos para hacer menús, etc. En el capítulo cuatro se hace una revisión de los manejos numéricos basados en funciones matemáticas sencillas y otras más complejas, como son las trigonométricas.

Los capítulos cinco y seis están dedicados al funcionamiento interno del *Amstrad* y los métodos disponibles para usar todas las facilidades. Puede verse el funcionamiento de los números binarios, así como del reloj, las interrupciones y el manejo de errores.

En el capítulo siete se explica otro tema que también posee profundo interés. Las estructuras de datos. Estas son fundamentales para todas aquellas tareas que precisen ficheros y en el libro se les da una amplia cobertura, recogiendo todos los sistemas de organización usados actualmente para estos menesteres y el modo de usarlos en el *Amstrad*. Esta descrición continúa en el capítulo 8, dedicado al proceso de datos.

Por último, y como es habitual, los dos últimos capítulos se dedican a dos temas que no podían faltar: los gráficos y el sonido. En ambos temas la información es abundante y se dan ejemplos de aplicación, como por ejemplo la realización de dibujos en tres dimensiones y control de la forma de onda del sonido.

Como es habitual en este tipo de libros, al final se incluye una serie de apéndices con los datos más habitualmente usados. El conjunto de caracteres *ASCII*, los códigos del teclado, los de color y los de error.

El libro trae una información muy extensa, y aunque en la presencia física deje algo que desear, resulta imprescindible para todos aquellos que quieran aprender a realizar manejo de ficheros y otros procesos por el estilo.



Los mejores programas del mes seleccionados por

AMSTRAD USER

PARTICIPE y GANE con nuestra revista uno de estos fabulosos premios



* 1 Impresora AMSTRAD DM-1
* 5 Lotes de 3 programas en cassette

- Para participar solamente deberá rellenar el cupón adjunto con los títulos de sus cinco programas favoritos en orden de preferencia y enviarlo a AMSTRAD USER.
- Todos los cupones recibidos antes del día 30 de cada mes entrarán en un sorteo
- A los premiados les será notificado por carta certificada en su domicilio.

AMSTRADIEZ Octubre 1985

Programas:

1— ....

2— ....

3— ....

4— ....

5— ....

Nombre: ....

Dirección: .... C.P. ....

Recorte y envíe este cupón a

AMSTRAD USER - AMSTRADIEZ

Avda. del Mediterráneo, 9
28007-Madrid

Una costumbre establecida desde hace tiempo en la industria del disco son las «listas de éxitos», en las que se da una relación de las canciones más vendidas. Con esta tabla intentamos algo parecido: daremos una lista de los programas mas vendidos, en definitiva los mejores, los diez puntos. Esperamos que sirvan de guía a la hora de decidir las compras.

AMSTRADIEZ

1 DECATHLON

Este juego, que nos permite emular a los héroes olímpicos, bate records. Se ve que el público español practica el deporte, ¡aunque sea sentado delante de la máquina!

2 FIGHTER PILOT

Los simuladores de vuelo han estado siempre en el punto de mira de los usuarios. Este gran programa basado en un F-18 ha supuesto un auténtico éxito en todos los ordenadores para los que existe.

3 GREMLINS

Otro éxito basado en una película taquillera. Estos animalillos horribles o simpáticos nos ofrecen la posibilidad de pasar un buen rato. Una aventura gráfico-conversacional.

4 PYJAMARANA

Más aventuras, en este caso de acción, con un argumento de pesadilla (y nunca mejor dicho, ya que se basa en los malos sueños de Wally). Un juego muy loco para usuarios muy locos.

5 KNIGHT LORE

Un precioso juego tridimensional en el que el protagonista es un hombre lobo. Un gran número de habitaciones, todas distintas. Para manejarlo bien hay que conjugar la inteligencia con la rapidez de reflejos.

Por último, queremos dar las gracias a las tiendas que han facilitado la información necesaria para confeccionar esta tabla:

Microtodo.
MicroWorld.
Sinclair Store.
WorldMicro.

¿Hay vida después del BASIC?



Hay más de un modo de pelar un pavo y probablemente cientos de modos de programar un ordenador. En este trabajo vamos a escribir sobre los lenguajes de programación, los otros, no el viejo y querido BASIC. Haremos, primero, una introducción de todos ellos desde un nivel muy genérico. También en este número hay un artículo sobre el LOGO que explica los conceptos básicos de este interesante lenguaje. En siguientes números veremos con más amplitud el resto.



Literalmente hay cientos de lenguajes de ordenador, algunos familiares y otros que nunca habrá oído hablar. Algunos de ellos son FORTRAN, RATFOR, ALGOL, ADA, MUMPS, FORTH, LOGO, PROLOG, APL, PL/11, LISP, COBOL, BASIC, C, VALID, SNOBOL, NEBULA, BLISS, MODULA-2 y SL-5. ¿Por qué?, se preguntaráusted, ¿hay tantos lenguajes? ¿Qué hay de malo con el BASIC?

La respuesta, en pocas palabras, es que no hay nada malo con el BASIC en sí, pero otros lenguajes pueden hacer algunas cosas mejor que el BASIC. A continuación explicamos por qué.

La **unidad central de proceso** *(Central Procesing Unit* o CPU) que forma el corazón de cada ordeandor no es nada más que un conjunto complejo de interruptores y casillas para almacenar datos temporales. La configuración de dichos interruptores en un momento dado se determina por la instrucción del programa que se está ejecutando y que llega a la **CPU.** Estas señales son impulsos eléctricos que representan ceros y unos (la presencia de tensión indica un 1, la ausencia es un 0. Aquí se entra en razonamientos filosóficos y no es la cuestión entrar en tanto detalle, pero esos ceros y unos, aunque manejados como números binarios por sencillez, son simples símbolos que son interpretados por el ordenador de una forma predeterminada. Las acciones que próvoca un determinado conjunto de símbolos están determinadas y son predecibles ya que han sido diseñadas en la **CPU** por sus creadores.

Cuando queremos que un ordenador nos haga el trabajo debemos darle los símbolos necesarios para que ejecute las instrucciones que nosotros queremos. Los conjuntos de números binarios no son precisamente los símbolos que mejor se recuerdan. He aquí un ejemplo: si se suministra el «número» 11101011 al Z80 (la CPU usada en el Amstrad) hace que se cambien entre sí los contenidos de dos registros internos de ésta. Si en cambio la combinación suministrada es 00111100, entonces el número almacenado en un registro temporal de la CPU denominado Acumulador (o registro A) se incrementa en 1, es decir, se le suma 1. Existen más de doscientas instrucciones con esta misma configuración que el Z80 es capaz de entender, podríamos decir que éste es el «lenguaje» que «habla»; de modo que todos los programa se presentan al final con este formato para que la **CPU** se ejecute.

Los símbolos o instrucciones que hacen que la CPU ejecute unas acciones determinadas se llaman «lenguaje máquina» cuando se unen para formar un programa. Resulta obvio que la palabra «máquina» resulta muy apropiada. A continuación se muestra un pequeño trozo de un programa en lenguaje máquina tal como lo ve la **CPU**.

```
00100011 | 11100110
01001111 | 00001111
01000111 |
```

¡Realmente no es muy comprensible!, pero es posible escribir el programa con otros símbolos más comprensibles y usar un programa para traducir estos símbolos en un código que puede entender la máquina. Como ejemplo se puede ver parte de un programa BASIC que utiliza símbolos bastante más fáciles de comprender (sobre todo si se entiende el inglés):

```
FORX=32 TO 127
PRINT CHR (X);" ";
NEXT X
```

Para el hombre medio de la calle puede que no sea mucho más informativo que la lista de números binarios que vimos antes. Pero cualquiera que haya programado alguna vez en BASIC reconocerá esto como parte de un programa que imprime los caracteres en la pantalla del ordenador. Algunos otros lenguajes se inclinan hacia el lado de la incomprensibilidad, algunos otros se parecen más al inglés ordinario. Aquí hay dos ejemplos de esto. El primero es parte de un programa en ensamblador.

```
JR      Z,ON2
LD      HL,REM1
CALL    TIME1
JR      TM13
BIT     1, (IX+REMO)
```

Este segundo ejemplo está escrito en PASCAL y es algo más fácil de seguir y se puede ver que está relacionado con puntuaciones y resultados (¿un juego más?).

```
BEGIN
 FOR I:=1 TO contador DO
  BEGIN
   IF puntos [I]<intento (media+12) THEN
     resultado:='BIEN'
   ELSE IF puntos [I]<intento (media-12)
     THEN
     resultado:='MALO'
   ELSE
     resultado:='NORMAL'
   END
END;
```

El lenguaje de su elección

El lenguaje ideal de programación sería el castellano. Imagínese que puediese teclear instrucciones del tipo:

LLEVA UN FICHERO CON EL BALANCE DE MI CUENTA BANCARIA Y QUE YO PUEDA INTRODUCIR EL IMPORTE DE CADA TALON JUNTO CON UNA FICHA QUE ME DIGA DE QUIEN ES Y POR QUE CONCEPTO. SI POR ALGUNA RAZON TECLEO LETRAS DONDE DEBE HABER NUMEROS, AVISAME CON UN PITIDO Y UN MENSAJE DE ERROR EN LA PANTALLA.

Desafortunadamente los lenguajes naturales (como es el castellano que hablamos) son demasiado variables e imprecisos para ser usados como instrucciones para un ordenador, de modo que ha sido necesario crear lenguajes más simples y rígidos de estructura. Debido a que cada persona utiliza los ordenadores para cosas distintas, los lenguajes han evolucionado siguiendo caminos distintos. Por ejemplo, el FORTRAM (uno de los primeros lenguajes) se escribió específicamente para simplificar la programación de expresiones matemáticas. El BASIC se hizo pensando en que fuese fácil de aprender y usar, lo que probablemente explica que sea tan popular. El PASCAL, otro lenguaje muy extendido, se desarrolló para enseñar a los estudiantes de informática las buenas técnicas de programación (se estudiará más el tema de las buenas técnicas de programación en el artículo sobre el PASCAL).

Autómatas, robots y máquinas que piensan

Algunos lenguajes se crearon en circunstancias especiales. El FORTH, por ejemplo, lo desarrolló el doctor Charles H. Moore para poder escribir programas que controlaran el movimiento de un telescopio. Su principio básico fue «hacerlo simple». Esto es un fragmento de un programa escrito en FORTH:

```
: CLEARSCREEN   12 EMIT;
: LEADIN        27 EMIT;
: CLREOS        LEADIN 74 EMIT;
: CLREOL        LEADIN 75 EMIT;
: HOME          LEADIN 72 EMIT;
: GOTOXY        LEADIN 61 EMIT SWAP 0
                MAX 79 MIN " EMIT
                0 MAX 23 MIN 32 " EMIT;
```

Si no parece simple se debe únicamente debido a que no se está familiarizado con él. Permite algunos trucos de programación y es usado en muchos de los juegos comerciales.

Otros lenguajes se escribieron específicamente para investigar el tema de la «inteligencia artificial». Entre ellos están el LISP, LOGO y PROLOG. Todos pertenecen a un grupo denominado de procesamiento de listas y son útiles no sólo como buenos lenguajes para aprender a programar, sino también para escribir programas que simulen comportamientos inteligentes y tomas de decisiones. Se incluyen dos ejemplos, el primero de ellos escrito en LISP:

```
—> (setq c (cons jam '(cdr x)))
—> (setq b (cons 'jam (cdr x)))
—> (setq a (reverse b))
—> (cdr (cdr a))
```

Y éste último está en LOGO, quizá el más famoso y extendido del grupo:

```
TO BOX
REPEAT 4 [FORWARD 50 RIGHT 90]
END
TO STAR
REPEAT 12 [BOX RIGHT 30]
END
```

Ventajas y fallos

El Amstrad es uno de los ordenadores más versátiles que existen. Además de un poderoso y rápido BASIC que viene con la máquina, existen otros disponibles opcionalmente. Por ejemplo, el LOGO y el PASCAL. También está en el mercado una versión ampliada del FORTH y está anunciada la aparición de «C», un lenguaje muy potente del que hablaremos en un futuro.

No es posible hablar de todos los puntos de estos lenguajes en este artículo, ya que justificar todas las afirmaciones lleva mucho espacio. No obstante, el siguiente resumen se puede tomar como válido.

BASIC

A favor: diseñado como un lenguaje para la enseñanza. Es muy fácil de aprender. Corregir errores es muy sencillo.

En contra: Puede favorecer malas técnicas de programación. Los programas se ejecutan lentamente y ocupan mucha memoria. Los distintos dialectos dificultan la compatibilidad.

PASCAL

A favor: Es un lenguaje «elegante» que favorece los hábitos limpios y lógicos de programación. Debido a que es un lenguaje compilado los programas se ejecutan muy rápido. También es autodocumentado (fácil de leer).

En contra: Es un lenguaje muy estricto e inflexible que no permite modificaciones de última hora. Bastante difícil de aprender. Corregir errores en los programas puede llevar mucho tiempo.

FORTH

A favor: El lenguaje es muy conciso y permite escribir programas concisos y compactos, ideal para sistemas con poca memoria. Muy flexible y permite añadir nuevos comandos al sistema en cualquier momento.

En contra: No es fácil de aprender y además no es muy claro de leer. Utiliza notación polaca inversa (RPN) y sólo números enteros.

LOGO

A favor: Diseñado en parte como un lenguaje educativo y en parte para estudiar la inteligencia artificial. Combina gran potencia con facilidad de uso (incluso un niño pequeño puede escribir programas con él).

En contra: Muy lento y por tanto no usable para escribir aplicaciones o software de sistema.

«C»

A favor: Un lenguaje rápido y potente que resulta ideal para escribir *software* de «sistema» como son sistemas operativos, compiladores, etc.

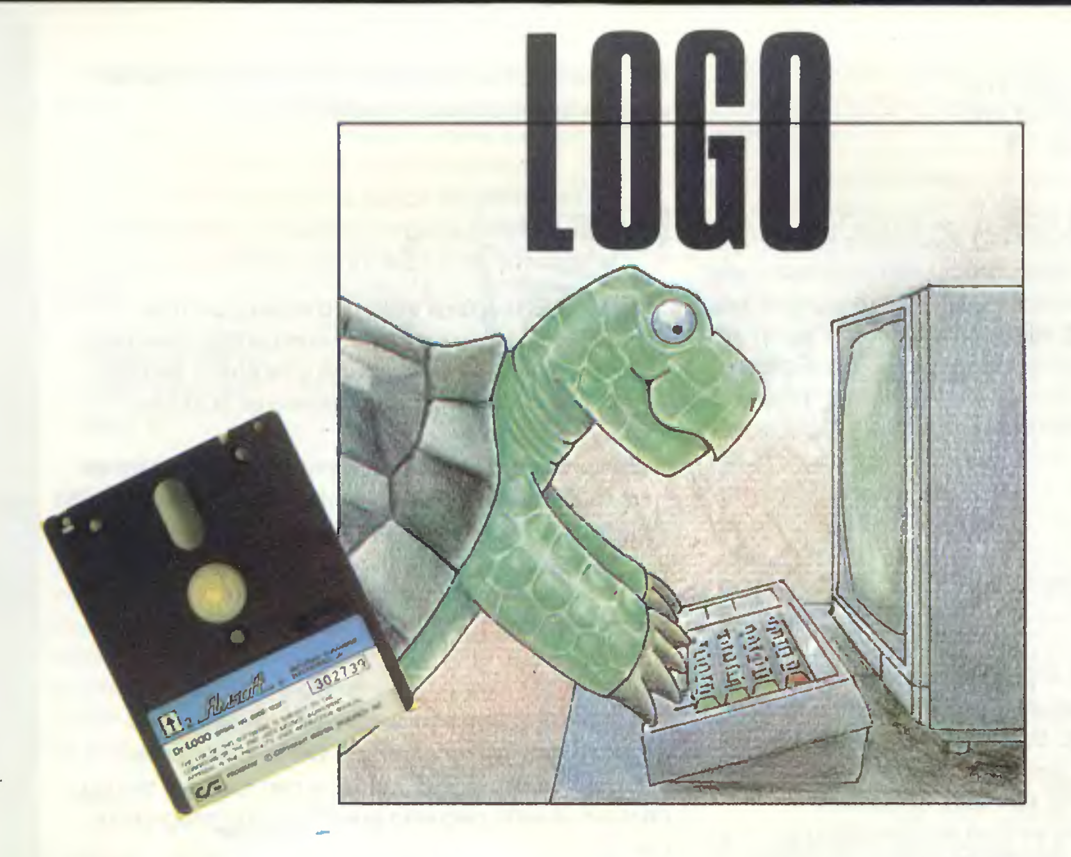
En contra: Es fácil cometer errores. Bastante difícil de aprender. No es indicado para principiantes.

El mejor para usted

Este pequeño recorrido por los lenguajes más populares le habrá convencido de que hay un lenguaje para cada tarea. Los programadores profesionales sabrán más de un lenguaje y como los buenos jugadores de golf, elegirán el adecuado para cada ocasión: El BASCI para los trabajos rápidos y «sucios», el FORTH donde se necesite memoria y velocidad, LOGO PARA la enseñanza, el PASCAL para los programas comerciales y muy largos y el «C» para las cosas realmente serias.



LOGO



Un lenguaje para la educaciòn

¿Por qué Amstrad ha incluido Dr. LOGO con la unidad de disco? ¿Qué puede hacer el usuario con LOGO que no pueda hacer con BASIC? ¿Por qué mucha gente que no tiene discos está ansiosa de tenerlos para poder usar Dr. LOGO? La razón más simple es que LOGO es uno de los lenguajes más potentes que existen para los microordenadores. Lo increíble es que se puede usar fácilmente incluso por los más jóvenes y es tan flexible que se usa para enseñar temas de inteligencia artificial a los graduados universitarios. El rango de las aplicaciones en que se puede usar el LOGO es realmente impresionante.

¿Está interesado en desarrollar aplicaciones tales como bases de datos?, el LOGO puede ser su lenguaje, ya que sistemas muy complejos de manejo de datos han sido desarrollados con este lenguaje y también se ha usado para producir un intérprete de un lenguaje de quinta generación, como es el **PROLOG.** También se utiliza para enseñar historia en las escuelas, así como otras múltiples aplicaciones.

Originalmente, el LOGO se desarrolló en el MIT **(Instituto de Tecnología de Massachusetts)** para trabajar con inteligencia artificial. La idea básica era desarrollar un lenguaje que fuese fácil de usar pero lo suficientemente potente como para desarrollar aplicaciones complejas. Inicialmente se dirigió a las escuelas primarias con la intención de ayudar a los niños en el uso del lenguaje y permitiendo el uso fácil de palabras y sentencias.

Muchas de las ideas básicas de este lenguaje se tomaron del LISP (en el glosario de la guía del usuario se habla más de este lenguaje). Una de las primeras aplicaciones que surgieron fue un sistema que traducía del inglés al «latín pobre», es decir, cambiar una frase por otra que no es latín pero que suena como si lo fuera. Lo que hace es coger las consonantes iniciales de la palabra y añadírselas al final junto con las letras «ay». Este tipo de texto codificado está basado en un tipo especial de jerga. Un ejemplo de la traducción cambia:

this is a sentence or so they tell me

a la siguiente versión:

isthay isay entencesay oray osay eythay elltay emay

Aunque es algo que se puede hacer en BASIC, el programa resultante es bastante más complejo que en LOGO.

También puede haber visto usted libros que tratan de enseñar inteligencia artificial usando BASIC, pero esto es como oír al grupo pop YES (de música sinfónica) en un tocadiscos portátil en lugar de un

estéreo en alta fidelidad. YES suena bien en mono, pero se necesita el estéreo para percibir todo el efecto. En lo que se refiere a la inteligencia artifical, el programar en BASIC es como programar en mono y necesita LOGO (o LISP) para programar en estéreo para obtener algo parecido a todo el efecto.

Aunque el LOGO se desarrolló para ayudar a los niños a usar un ordeandor, es más famoso ahora por el uso de los gráficos de tortuga **(Turtlegraphics).** Estos gráficos fueron creados por el profesor Seymour Papert (también del MIT) como un medio interesante de éxplorar otros aspectos de la programación. El resultado tuvo tanto éxito que se implementaron en muchos otros lenguajes como son algunas versiones del PASCAL. La idea de los gráficos de tortuga es muy sencilla.

Piense que tiene que darle instrucciones a alguien para que llegue a un sitio: «siga tres kilómetros, gire a la derecha, siga otros seis kilómetros y gire a la izquirda. Está a dos kilómetros del cruce». En el LOGO puedes dibujar formas dando instrucciones de este modo que son ejecutadas por algo llamado «tortuga». Algunas tortugas son robots que se mueven por el suelo, pero la mayoría de las veces son dibujos en la pantalla del ordenador. Aunque existe, al menos, una versión del LOGO que utiliza un dibujo que se parece a una tortuga, la versión del **Amstrad** utiliza una flecha ya que es más útil que la tortuga debido a que se ve fácilmente la dirección a la que apunta.

Para hacer que dibuje un cuadrado en la pantalla, por ejemplo, usamos las órdenes

```
fd 50 rt 90
fd 50 rt 90
fd 50 rt 90
fd 50 rt 90
```

Estas órdenes son fáciles de entender si se sabe un poco de inglés. fd significa «forward» (adelante) y rt es una abreviatura de «right» (derecha). La tortuga se mueve 50 unidades hacia adelante y gira 90 grados, después de lo cual se repite la secuencia otras tres veces. Si quiere comprobarlo figúrese que es una tortuga y ande por el suelo de su casa siguiendo estas órdenes.

Otra forma de hacer lo mismo es

```
repeat 4[fd 50 rt90]
```

que es autoexplicativa si se sabe que repeat significa «repite». Si existe una variable que se llama, por ejemplo, «lado», podemos usarla para decirle a la tortuga cuánto queremos que avance en cada lado y, por tanto, de qué tamaño queremos el cuadrado:

```
repeat 4[fd : lado rt 90]
```

donde los dos puntos delante de la variable indican que nos estamos refiriendo a su valor. El dibujo de cuadrados se puede automatizar en gran medida creando un procedimiento (que con gran originalidad vamos a llamar cuadrado).

```
to cuadrado : lado
repeat 4[fd : lado rt 90]
end
```

Donde :lado es un dato de entrada (o parámetro) que nos permite elegir el tamaño del lado del cuadrado.

Una vez hecho esto, y si quisiésemos dibujar un cuadrado de lado 50, sólo tendríamos que introducir

```
cuadrado 50
```

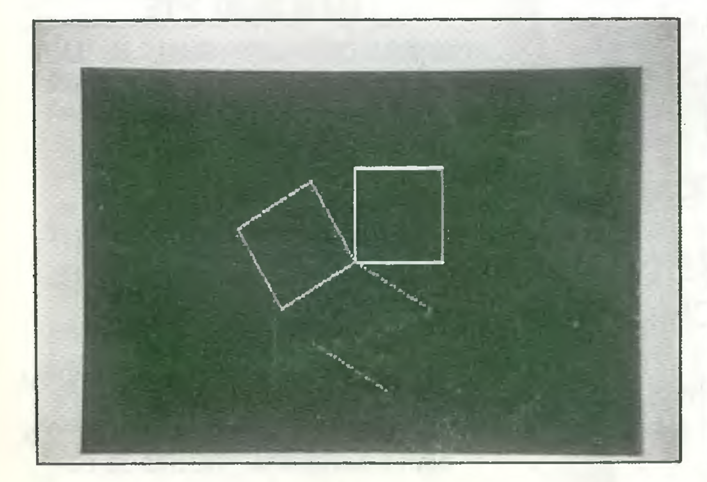
que nos dibujará automátiamente un cuadrado de cincuenta unidades de lado. Al terminar, la tortuga se nos queda en la misma dirección que estaba al principio y si introducimos rt 58, girará 58 grados a la derecha. Si por último le decimos

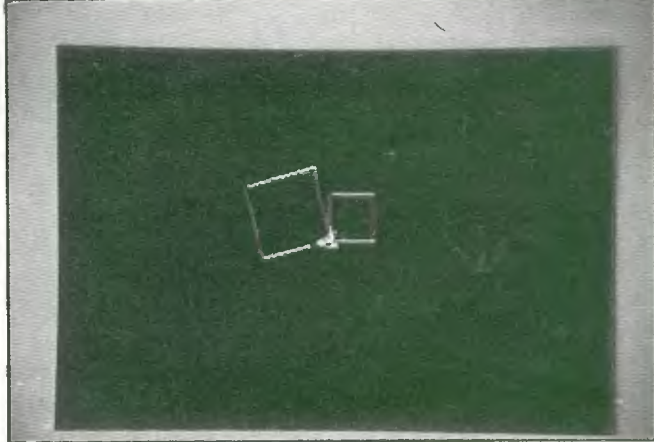
```
cuadrado 100
```

nos dibujará un cuadrado de 100 de lado girado con respecto al primero.

La figura 1 muestra el resultado de la secuencia

```
cuadrado 50 rt 58
cuadrado 100 lt 162
cuadrado 80
```





y si mira cuidadosamente podrá ver la flecha que representa a la «tortuga». Esta es del mismo color que el último cuadrado dibujado (el de la izquierda) y se dará cuenta que cada uno es de un color distinto. La diferencia en colores se debe a un ligero cambio en el procedimiento cuadrado. La nueva forma es:

```
to cuadrado :lado
 setpc 1+resto (item 5 tf) 3
repeat 4[fd :lado rt 90]
end
```

Donde la frase que empieza con setpc es nueva. Todas las líneas se pueden dibujar en tres colores distintos (incluido el del fondo) y cada uno de los tres corresponde a un «rotulador» —pen— numerados del 1 al 3. La línea que se ha añadido al procedimiento cambia el color en la secuencia 1,2,3,1,2... cada vez que dibujamos un cuadrado.

El elemento (ítem) quinto de la lista de datos de la tortuga (tf) es el número actual de color y el resto se obtiene cuando ese número se divide por tres. El procedimiento resto se define como

```
to resto :p1 :p2
op :p1 – (int (:p1 /
   :p2) * :p2)
end
```

que devuelve (op) el valor mostrado. El procedimiento resto es similar a una función operador del BASIC, ya que una función también devuelve un valor y en realidad el procedimiento resto corresponde a la función *MOD* del BASIC. Incluso con simples cuadrados es posible obtener efectos intrigantes como se puede ver en las figuras 2 ó 3 que sólo usan cuadrados.

En la figura 2 se puede ver fácilmente cómo están relacionados los cuadrados. El procedimiento usado fue:

```
to demo2 :lado
cs fs ht
repeat 3 [cuadrado :lado
        lt 120]
end
```

Que es sencilla de comprender aparte de *«cs fs ht»*. Esta línea le dice al ordenador «borra la pantalla, ponla toda en modo gráfico y oculta la tortuga». La figura 3 parece mucho más compleja pero está construida únicamente con cuadrados y la línea clave es

```
cuadrado :lado fd
         :lado rt :ángulo
```

Y para obtener ese dibujo se fueron incrementando los lados en 3 y el ángulo es 117.

Una característica importante del LOGO se muestra en la figura 4, que es un gusano. Para hacer esto se gira un ángulo aleatorio, se hace un movimiento aleatorio, se coge un color aleatorio del rotulador y se vuelve a llamar a gusano (que significa hacer otro giro aleatorio, otro movimiento aleatorio...)

```
to gusano
rt random 360 fd random 50
setpc 1+random 3
gusano
end
```

y antes de empezar a dibujar hay que decir *wrap gusano*, ya que esto significa que las líneas pasan de un lado al otro de la pantalla sin desaparecer (como sucede con el modo normal o de ventana). La habilidad de un procedimiento para llamarse a sí mismo se denomina «recursión» y puede ser muy útil.

Un ejemplo intrigante de la recursión es el dibujo del árbol de la figura 5. El procedimiento usado se llama árbol y damos el listado por si alguien lo quiere probar, pero no se preocupe si no lo entiende, sólo está aquí como referencia.

```
to árbol :longitud :ángulo :orden :factor1 :factor2
ht pd
if :orden=0 [stop]
1t :ángulo fd :longitud
árbol :longitud * :factor1 :ángulo :orden–1
:factor1 :factor2
pu bk :longitud rt 2 * :ángulo pd
fd :longitud * :factor1 :ángulo :orden–1
:factor1 :factor2
pubk :longitud * :factor2 1t :ángulo pd
end
```

La figura se ha hecho con la llamada

```
árbol 90 35 9 0.8 0.4
```

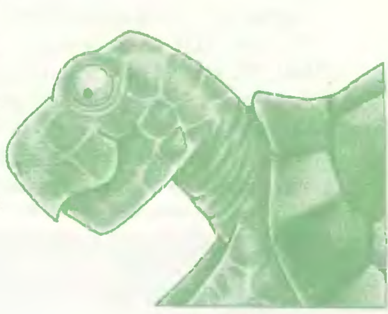
y para ayudarle a comprender cómo se hace el árbol, se puede usar un ejemplo más sencillo, como es

```
árbol 150 35 3 0.8 0.4
```

Una ayuda en este aspecto es que el árbol empieza desde la parte más inferior y el dibujo de las líneas va (incluso retrocediendo por el camino hecho) desde la izquierda a la derecha.

Cuando investigue con este procedimiento recuerde que el LOGO no sólo puede producir dibujos complejos, sino también expresiones verbales o numéricas de la misma complejidad. Aquí se han usado los gráficos porque son más fáciles de ver, pero no olvide que el LOGO es capaz de mucho más.

Cuando investigue con este procedimiento recuerde que el LOGO no sólo puede producir dibujos complejos, sino también expresiones verbales o numéricas de la misma complejidad. Aquí se han usado los gráficos porque son más fáciles de ver, pero no olvide que el Logo es capaz de mucho más.



CORREO



¿Se pueden pasar programas comerciales de cinta a disco?

Un programa comercial en cinta, en general, no se puede pasar a disco, puesto que el programa está protegido y no se puede detener para dar las instrucciones de grabar.

Los programas de utilidades del CP/M llamados CLOAD Y CSAVE, que pasan programas de cinta a disco y viceversa, exigen como condición que los programas o ficheros estén en formato ASCII.



¿Cómo se carga un fichero secuencial ya creado?

La estructura general es:

— Dimensionar una matriz con el número máximo de elementos a cargar.

— Abrir el fichero (OPENIN «nombre d fichero»).

— Bucle que carga en la matriz los datos (la instrucción para leer un dato del cassette o del disco es «INPUT#9», nombre).

— Cerrar el fichero (CLOSEIN).

El error «EOF met» (encontrado FINAL DE FICHERO) se produce al querer leer los datos cuando ya no hay más, habiéndose alcanzado la marca de final de fichero. Este error se puede evitar sabiendo el número exacto de datos que vamos a leer o usando la instrucción EOF.

```
10 DIM A$(5)
20 OPENIN "COSAS"
30 IF EOF THEN 70
40 INPUT#9,A$(I)
50 PRINT A$(I)
60 GOTO 30
70 PRINT "Final de Fichero":CLOSEIN
```

Los datos no se leen directamente uno a uno del cassette o del disco, sino que más bien se carga un bloque de información que suele tener 2 K (o menos, en caso de ser el último bloque y no estar completo hasta las 2 K) en un BUFFER o memoria auxiliar. La instrucción «INPUT#9, nombre» lee, por lo tanto, de este BUFFER. Al leer un dato que no está en el BUFFER, sino a continuación en la cinta o el disco, el ordenador vuelve a cargar un nuevo bloque.



¿Cómo se crea un fichero secuencial sencillo?

En general, los pasos a seguir para crear un fichero son:

1. Dimensionar una matriz.
2. Introducir datos en los elementos de la matriz.
3. Grabar los datos, de la memoria a una cinta o disco.

Las instrucciones de ficheros estarían en el tercer apartado e incluirían SIEMPRE:

— Instrucción OPENOUT «nombre de fichero» (Abrir el fichero).

— Un bucle que contenga la instrucción PRINT#9, nombre o WRITEN#9, nombre (Aquí es dónde se graba «secuencialmente» toda la información que queremos almacenar).

— Instrucción CLOSEOUT (Cerrar el fichero).

Ejemplo:

```
10 DIM A$(5)
20 A$(1)="LIBROS":A$(2)="DISCOS":
A$(3)="CINTAS":A$(3)=
"PROGRAMAS":A$(4)="REVISTAS":
A$(5)="ORDENADOR"
30 OPENOUT "COSAS"
40 FOR I=1 TO 5
50 WRITEN#9,A$(I)
60 NEXT I
70 CLOSEOUT
80 END
```

Cuando se da la orden «PRINT#9, nombre» o «WRITEN#9, nombre» los datos no se graban inmediatamente en la cinta o el disco, sino que se almacenan temporalmente en una zona de memoria llamada BUFFER. Este BUFFER se vuelca en el cassette o en el disco, una vez lleno, de una manera automática, o bien, nosotros podemos forzarlo al dar la instrucción CLOSEOUT.



¿Qué capacidad tiene los discos de Amstrad?

Los discos de AMSTRAD de 3 tienen una capacidad de 256 K por cada cara sin formatear.

Características en todos los formatos:

— Simple cara (cada cara es tratada independientemente).

— 512 bytes por sector.

— 40 pistas (numeradas de 0 a 39).

— 1024 bytes por bloque (en CP/M).

Al formatear obtenemos las siguientes capacidades:

— FORMAT V o S 169 K por cada cara.

— FORMAT D 178 K por cada cara.

— FORMAT I 154 K por cada cara. (sólo para uso profesional).

Esta sección está dedicada a solucionar las dudas que los usuarios tengan con el ordenador. Si usted tiene algún problema, no dude en consultarnos:

Estamos en:

Amstrad User

C/ Bravo Murillo 377, 5.° a.

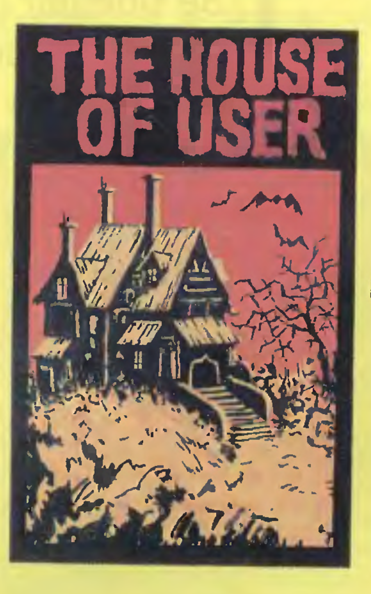
28020

AMSTRAD USER

OFERTA ESPECIAL DE SUSCRIPCION

La revista Amstrad User te ofrece la oportunidad de suscribirte en unas condiciones muy ventajosas:

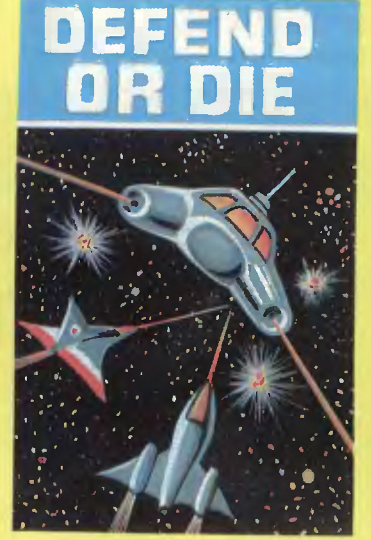
1. Recibir, directamente en tu domicilio, una revista mensual que durante 12 meses te proporcionará interesantes artículos, programas, sugerencias, etc. En definitiva, sacar el mayor rendimiento posible a tu ordenador.
2. Conseguir, totalmente gratis, dos entretenidos programas cuyo valor comercial es de **3.800** ptas.



THE HOUSE OF USER

El juego se desarrolla en una casa embrujada y nuestra emoción es el descubrir el secreto que en ella se encierra

EMOCION Y ENTRETENIMIENTO ASEGURADO



DEFEND OR DIE

Batalla galáctica con radar de posición.

3. Beneficiarse de un ahorro de **500** ptas. sobre el precio normal de suscripción.
4. La opción de ser protagonista. Puedes tener una participación directa en **Amstrad User** con tus comentarios, sugerencias, etc.

NO DEJES PASAR ESTA FABULOSA OPORTUNIDAD

Envía el cupón adjunto, que no necesita franqueo, hoy mismo.

CORREO



¿Cómo se define el carácter «Ñ»?

Para definir nuevos caracteres debes usar la instrucción SYMBOL. Esta instrucción lo que hace realmente es redefinir un carácter de los 256 que vienen en el código ASCII, reemplazándolo por el nuevo (para mayor información consulta en el manual del CPC464 C1.P14 y C8.P59 y en el del CPC664 3/80 y 8/20). Vamos a utilizar, por ejemplo, los caracteres 255 y 254 para definir la «Ñ» minúscula y mayúscula, respectivamente, Teclea:
SYMBOL 255,60,0,216,102,102,102, 102,0 «minúscula»
SYMBOL 254,218, 230,246,222,206,198,198,0 «MAYÚSCULA» Ahora si quieres imprimir o ver en la pantalla las dos «Ñs» escribe:
PRINT chr$(255);chr$(254)

Con COPY y las teclas del cursor se pueden copiar los caracteres visibles en la pantalla e introducirlos en una sentencia. También se pueden usar como chr$(255) y chr$(254), respectivamente. Por ejemplo:
nombre$="Espa"+chr$(255)+"a":
PRINT nombre$

La «Ñ» se puede colocar en una de las teclas -con la instrucción KEY DEF- para que al pulsar salga directamente por la pantalla. En este caso sólo puedes usar los caracteres del 32 al 127 o del 160 al 223 (para más información te remitimos al MANUAL DE REFERENCIA BASIC para el programador AMSTRAD CPC464 pag. K.2). Vamos a realizar un ejemplo de esto:

```
10 SYMBOL AFTER 222
20 SYMBOL 222,60,0,216,102,102,
102,102,0
30 SYMBOL 223,218,230,246,222,
206,198,198,0
40 KEY DEF 22,1,222,223
```

En este caso hemos incluido las dos «Ñs» en la tecla «/», situada al lado de la tecla SHIFT de la derecha. Si pulsas la tecla, después de haber ejecutado el programa anterior se escribirá en la pantalla la «ñ», y pulsando SHIFT simultaneamente aparecerá «Ñ».

Cuando menor sea el número del SYMBOL AFTER mayor cantidad de memoria se emplea, por lo tanto para ahorrar memoria calcula el número total de caracteres que vas a usar para saber qué número debes poner. Por ejemplo, el número mayor que podemos usar es 223, por lo ya comentado más arriba, pero necesitamos dos caracteres: uno para la minúscula (el 223) y otro para la mayúscula (el 222); por lo tanto el valor que debemos utilizar en el SYMBOL AFTER es 222.

Recuerda incluir estas instrucciones al principio de tu programa, en el caso de que preveas que puedes necesitar la «Ñ».



¿Son compatibles los modelos Amstrad CPC464 y CPC664?

Todos los programas realizados para el CPC464 sirven para el CPC664, con la excepción de un programa que utilice la zona de memoria reservada para la unidad de disco situada en la parte superior de la RAM. Esta zona se encuentra a partir de la dirección 42620 y ocupa 1284 bytes. Se utiliza para la tabla de comandos externos (instrucciones de disco) y para la parte de RAM que necesitan las ampliaciones de ROM.

Los programas del CPC664, hechos en BASIC, serán compatibles con los del CPC464 siempre y cuando no utilicen las instrucciones nuevas implementadas en el nuevo modelo, que son:

— CLEAR INPUT
— COPYCHR$
— CURSOR
— DEC$
— DERR
— FILL
— FRAME
— GRAPHICS PAPER
— GRAPHICS PEN
— MASK
— ON BREAK CONT

La razón por la que muchos programas en cinta no cargan en el CPC664, no es problema de compatibilidad, sino del cassete. Esto se soluciona ajustando el volumen correctamente o, en caso necesario, el tornillo de ajuste del cabezal.



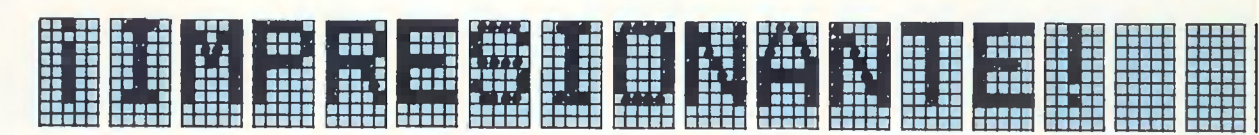
¿Se pueden usar ficheros de acceso aleatorio con una unidad de disco?

El ordenador AMSTRAD en sus diferentes modelos no preveía ficheros de acceso directo desde BASIC. Sin embargo, INDESCOMP ha diseñado una rutina llamada RANDOM-FILES que se obsequia en una cinta al comprar la unidad de disco o dentro del disco de regalo en la compra del CPC664.

Esta rutina crea unas instrucciones de BASIC típicas de ficheros aleatorios: abrir o cerrar fichero, y leer o grabar aleatoriamente un registro del fichero, con lo que este problema se soluciona.

Esta sección está dedicada a solucionar las dudas que los usuarios tengan con el ordenador. Si usted tiene algún problema, no dude en consultarnos:
Estamos en:
Amstrad User
C/ Bravo Murillo 377, 5.° a.
28020 Madrid

ASTRON 1400



Impresionante en precio y prestaciones. Con todo lo que puede y debe exigirse a una impresora de la más avanzada tecnología: alta velocidad (140 caracteres por segundo), amplio juego de caracteres, alimentación con cualquier tipo de papel (continuo, hojas sueltas, en rollo), excelente calidad de impresión (matriz de 9 × 9), interface centronics (100% compatible con el IBM PC) y, opcionalmente, Serie RS-232 C, ...

Compacta y robusta hasta en el menor detalle de diseño y a un precio que hasta hoy parecía imposible. Impresionante ASTRON 1400.

ESPECIFICACIONES

Sistema: Matriz de puntos (9 × 9) con agujas reemplazables.

Velocidad: 140 c.p.s.

Fuentes: 96 caracteres ASCII y 32 semigráficos.

Columnas: 80 (caracteres normales), 132 (comprimidos), 40 (ensanchados), 66 (ensanchados-comprimidos).

Alimentación: Fricción y tracción.

Papel: Hojas sueltas de 102 a 254 mm. de anchura.
Papel continuo de 242 mm. (opcionalmente de 102 a 254 mm.).
Papel en rollo de 102 a 254 mm.

Interface: Paralelo CENTRONICS. Opcionalmente, Serie RS-232C.

Buffer: 2 kbytes de memoria.

P.V.P. **66.900 pts.**

